University of Zurich[UZH]

Department of Informatics

Refinement and optimization methods for reconstructing and modeling indoor environments

Dissertation submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by
Georgios-Tsampikos Michailidis
from Greece

approved in September 2018

at the request of
Prof. Dr. Renato Pajarola
Prof. Dr. Reinhard Klein

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, September 25, 2018.

The Chairman of the Doctoral Board: Prof. Dr. Sven Seuken.

ABSTRACT

Driven by the development in digital 3D imaging and scanning technology, the efficient capturing and modeling of 3D objects and environments has become a critical task in many application domains such as in architecture, engineering, robotics, navigation, construction and facility management. In particular, the availability of highly expressive 3D models from indoor environments could create highly added value in these domains, since these models, generally known as *Building Information Models* (BIMs), could provide semantically rich representations of the scene, allowing its analysis and manipulation. Also, they could provide accurate identification of their main architectural wall structures, such as the windows and doors, based on their *as-is* condition, which might be significantly different from the one they were designed.

Despite the many recent research efforts, the fully automatic generation of semantically enriched 3D models from building interiors still remains a very challenging and time-consuming process. The main difficulties lie on the accurate and fast acquisition of the surrounding environment and the creation of a faithful and semantically rich 3D model. Although the open problems in the field are still manifold, we focused in this thesis specifically on the following three important challenges: first, how to capture efficiently the 3D information of the scene, using a method which combines increased accuracy, high performance and produces adequate results for the majority of real-world indoor applications; second, how to faithfully capture the finer details of structural building elements and generate a semantically rich building model; third, how to allow the successful and auto-

matic reconstruction of 3D BIM models without relying on restrictive assumptions about the scene or the dataset.

For each of these issues, we propose a solution which advances the state-of-the-art and allows for improved performance and better quality to the extracted results. Our first work focuses on the efficient and fast acquisition of the scene and introduces a new hardware-efficient stereo vision method. In our method, a local-based correlation algorithm computes the matching cost values and an optimization technique based on Discrete Dynamical Systems refines the extracted depth information. In the same contribution, we propose also an efficient parallel-pipelined hardware architecture, which implements the proposed stereo reconstruction method on a custom FPGA device, allowing high processing speeds for high-resolution stereo images.

The second research contribution of this thesis focuses on the 3D modeling stage of the reconstruction pipeline and aims to recover and semantically label the architectural wall elements of indoor environments. This is achieved by partitioning the wall surfaces of the reconstructed building models into small planar patches and classifying them using a bayesian graph-cut optimization technique. Due to its beneficial design, our approach can be embedded as a post-processing unit to the majority of modern modeling pipelines, enabling them to extract automatically semantically rich 3D models.

The last research contribution of this thesis lifts a restrictive and widely-used assumption in the field, that the scanner viewpoint positions should be known a priori, in order the reconstruction and 3D modeling of the scene to be performed successfully. Specifically, this work constitutes the first method for re-engineering and reconstructing the original scanner positions from raw point clouds. It relies on the scanning characteristics of the acquisition process and employs a statistical analysis to raw point data, in order to reveal the features which will allow the retrieval of the true scanner positions.

Extensive qualitative and quantitative evaluations on real-world and synthetic datasets reveal the advantageous behavior of the proposed methods, as well as their efficiency and performance under challenging and difficult indoor conditions.

KURZFASSUNG

Angetrieben durch die Entwicklung der digitalen 3D-Bild- und Scantechnologie ist das effiziente Erfassen und Modellieren von 3D-Objekten und -Umgebungen in vielen Anwendungsdomänen wie Architektur, Ingenieurwesen, Robotik, Navigation, Konstruktion und Facility Management zu einer kritischen Aufgabe geworden. Diese Modelle, die allgemein als Building Text Models (BIMs) bekannt sind, liefern aussagekräftige Darstellungen von Szenen und ermöglichen Analyse und Manipulation der darunterlegenden Daten. Sie ermöglichen auch eine genaue Identifizierung ihrer wichtigsten architektonischen Elemente, wie Fenster und Türen, auf der Grundlage ihres derzeitigen Zustandes.

Trotz der vielen jüngsten Forschungsanstrengungen, bleibt die vollautomatischen Erzeugung von ausreichend ausdrucksstarken und semantisch angereicherten 3D-Modellen von Innenräumen eine grosse Herausforderung und zeitaufwendig. Die Hauptherausforderungen in diesem Prozess liegen in der genauen und schnellen Erfassung der Umgebung und der Erstellung eines echten und semantisch reichen 3D-Modells.

Obwohl die offenen Probleme auf dem Gebiet immer noch vielfältig sind, haben wir uns in dieser Arbeit auf die folgenden drei wichtigen Herausforderungen konzentriert: Erstens, die Erfassung und Kombination von 3D-Informationen einer Szene mit Methoden, die erhöhte Genauigkeit, hohe Leistung und angemessene Ergebnisse für die meisten professionellen Indoor-Anwendungen bieten; zweitens, die Erfassung feinster Details in strukturellen Bauelementen und automatische Erstellung eines semantisch reiches Gebäudemodells; drittens, die erfol-

greiche automatische Rekonstruktion von 3D-BIM-Modellen, ohne auf restriktive Annahmen über die Szene oder den Datensatz angewiesen zu sein.

Für jedes dieser Probleme schlagen wir eine Lösung vor, die den Stand der Technik voranbringt und verbesserte Ergebnisse und eine bessere Qualität liefert. Der erste Teil dieser Arbeit konzentriert sich auf die effiziente und schnelle Erfassung einer 3D Szene und stellt eine neue Hardware-effiziente Stereo-Vision-Methode vor. Die Methode benutzt einen lokalen Korrelationsalgorithmus kombiniert mit einer Optimierungstechnik, welche auf sogenannten "diskret dynamischen Systemen" zur Verfeinerung von Tiefeninformation basiert. Im selben Beitrag schlagen wir eine effiziente parallel Hardware Architektur vor, welche die vorgeschlagene Stereorekonstruktion auf einem kundenspezifischen FPGA-Gerät implementiert und hohe Verarbeitungsgeschwindigkeiten für hochauflösende Stereobilder ermöglicht.

Der zweite Teil dieser Dissertation konzentriert sich auf die 3D-Modellierung während der Rekonstruktionsphase. Das Ziel ist die Klassifikation von kleinen ebenen Teilen in beliebigen architektonischen Wandflächen mit einer sogenannten Bayes-Graph-Cut-Optimierungstechnik. Die Klassifikation und Erkennung solcher Wandflächen in beliebigen Innenräumen ermöglicht die automatische Extraktion von sematischer Information von 3D-Modellen. Aufgrund seines vorteilhaften Designs kann unser Ansatz als Nachbearbeitungsschritt in die meisten modernen 3D-Modellierungs Workflows integriert werden.

Der letzte Forschungsbeitrag dieser Arbeit hebt eine restriktive und weit verbreitete Annahme auf, dass die Laserscanner Positionen a priori bekannt zur Rekonstruktion und 3D-Modellierung der Umgebung sein müssen. Diese Arbeit präsentiert die erste Methode zur Re-Engineering und Rekonstruktion der originalen Laserscanner Positionen aus rohen Punktwolken. Unser Ansatz beruht auf den intrinsischen Scan- Eigenschaften des Akquisitionsprozesses und verwendet eine erweiterte statistische Analyse für rohe Punktdaten, die die Merkmale aufzeigt, die das Abrufen echter Scannerpositionen ermöglichen.

Umfangreiche qualitative und quantitative Bewertungen von realen und synthetischen Datensätzen zeigen die Vorteile der vorgeschlagenen Methoden sowie deren Effizienz und Leistung unter den anspruchsvollen Bedingungen in Innenräumen.

ACKNOWLEDGMENTS

The work presented in this thesis is originated from the contribution and support of many people all these years.

First and foremost, I would like to express my gratitude to my advisor, Prof. Dr. Renato Pajarola, for this wonderful opportunity he gave me to pursue this doctorate at the Visualization and Multimedia Lab (VMML) at the University of Zurich. Among other things, I am grateful for his constant support and patience, and I consider myself very lucky for being able to benefit from his knowledge, experience and personality. I also thank him for his numerous advices, guidance and understanding he showed on many occasions all these years.

I would like also to thank Prof. Dr. Ioannis Andreadis for his help and advices, especially during the development of the first contribution of this thesis. I express also my gratitude to Prof. Dr. Reinhard Klein for being the co-advisor of my thesis. I deeply thank also Prof. fr. Georgios Anagnostopoulos for his constant help and support in my personal and research life, but especially I thank him for being the driving force to initiate my PhD studies.

Furthermore, I would like to thank all former and current colleagues at VMML for their support during my research, while special thanks go to Matthias Thöny for his valuable help.

From the bottom of my heart, I would like also to thank Her, my mother and father for their unlimited love and for being always there for me. Last but not least, I would like to deeply thank my wife and children for their unconditional moral support, love and patience during all these years.

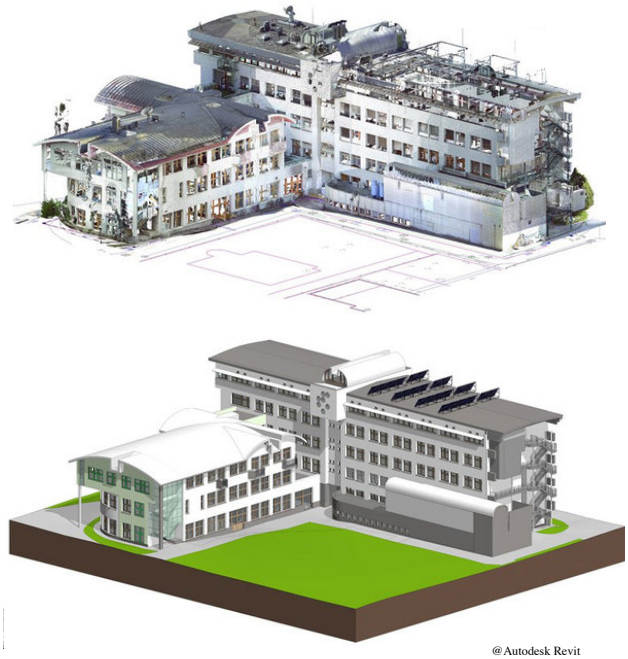
CONTENTS

LIST OF FIGURES

LIST OF TABLES

C H A P T E R 1

INTRODUCTION



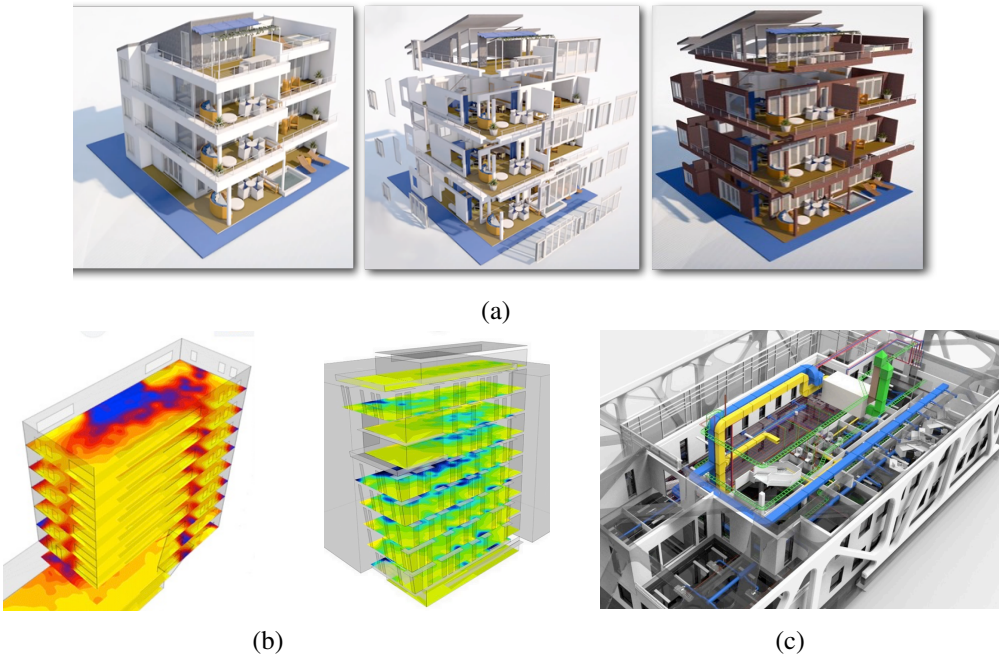
@Autodesk Revit

1.1 Introduction and Motivation

During the last years, new advances in 3D geometry acquisition technologies have allowed significant improvements in capturing, processing and reconstruction of objects and environments, opening new horizons to many fields of science and engineering. Among these fields, high attention concentrates the automatic modeling of building environments, mainly due to the importance buildings have in our everyday life, but also due to the variety of applications that could benefit from the availability of accurate and semantically rich digital building representations.

Until recently, the research community has focused mainly on the automatic reconstruction of outer building shapes [Stamos and Allen, 2000; Pu and Vosselman, 2009]. Although the 3D modeling of building façades can be beneficial to various fields such as urban planning, heritage documentation and computer games, it provides only a limited overview of the information we can derive from building models. Considering the details included in indoor environments, we can enrich these models with additional valuable information and broaden their application areas beyond the ones covered by the reconstruction of building exteriors.

In particular, models from building interiors could create highly added value in the Architecture, Engineering, and Construction (AEC) domain, where there is an increasing demand for 3D *Building Information Models* (BIMs), that is, semantically rich building representations that faithfully capture the condition in which



(a) (b) (c)

Figure 1.1: *BIM model examples used in property management [ArchiCAD, 2014] (a), in facility energy analysis [bimen, 2015] (b) and in engineering [adi, 2015] (c).*

the buildings or structures are actually constructed (the "*as-is*" condition), as opposed to the way they were planned [Volk et al., 2014]. Such precise 3D models focus mainly on the accurate reconstruction of permanent building structures (e.g. walls, ceiling and floor) and their architectural wall elements (e.g. windows and doors). Therefore, they can be natively applied in various applications, ranging from visualization or simulation to robotics, navigation, construction and facility management (see also Figure 1.2). Furthermore, they could help interior designers for planning renovations, while they could be also used in gaming industry for representing realistically the games' environments.

In practice, the creation of 3D models from building interiors consists of an intensive two-stage reconstruction process, which includes the *data acquisition* and *3D modeling* of the scene. In the first stage, the 3D information of the physical scene is captured, either using passive (e.g. stereo vision and photogrammetric) or active (e.g. terrestrial laser scanners) range sensors [Strecha et al., 2007; Kaller et al., 2016]. Especially for reconstructing the building environments, laser scanners and stereo vision systems constitute the most efficient and widely used methods in the field [Strecha et al., 2008; Zhu and Leow, 2013]. Proceeding to the second reconstruction stage, the derived range data usually form a 3D point cloud, whose 3D points are classified and segmented to form large arrangements that constitute the permanent structures of the scene (e.g. walls). By further enriching these components with additional semantic information about their wall elements (windows and doors), the digital 3D model of the building is derived.

In spite, however, of the initial research efforts for generating faithful building representations, the fully automatic semantic 3D reconstruction of building interiors still remains a cumbersome, challenging and time-consuming process, while it requires most of the times human intervention for its successful completion [Xuehan et al., 2013; Volk et al., 2014]. Therefore, there is a strong need to optimize and refine the stages of the aforementioned reconstruction process, reducing or avoiding the restrictions introduced by the current methods and real-world requirements. Thus, in this thesis we perform targeted interventions to the reconstruction process and propose methods that improve both the acquisition and modeling stages. Emphasizing also on their generalization and broad applicability, the introduced methods can be applied off the shelf to the majority of current reconstruction and modeling pipelines, refining their workflows and providing immediate improvements to the extracted results.

1.2 State of Research

The problem of reconstruction and semantization of indoor environments is a multifaceted problem that has been studied a lot the last few years. Given the range

of literature in this field, we restrict our analysis mainly to the approaches that are closest in scope to our work. Thus, in this section, we provide an initial, general overview of the methods that outline the general context under which our work will be placed, while a more focused and detailed analysis is provided in the next chapters (in particular, in Sections 3.1.2, 4.1.2 and 5.1.2).

A. Scene Measurement and Data Acquisition

As already stated in the previous section, the process of measuring the scene of interest can be performed either by using active techniques that rely on terrestrial laser scanners, or by using passive methods such as stereo vision. As outlined in [Besl, 1988; El-Hakim et al., 1995; Sansoni et al., 2009], terrestrial laser scanners and LiDAR (Light Detection And Ranging) devices can measure accurately objects in long ranges under various light conditions, although their acquisitions are influenced by high-reflectance surfaces and sharp discontinuities. In addition, when multiple 3D point clouds are necessary to be captured in order to represent faithfully the geometry of the scene, they need in a later stage to be registered to a common coordinate system [Gielsdorf et al., 2004; Rabbani et al., 2007]. This is, however, a time-consuming process which is mainly performed manually, and along with the increased scan time and the high cost of LiDAR devices, they comprise often prohibitive factors for using laser scanners in specific applications.

An alternative solution for acquiring the 3D information of the scene constitutes the use of passive systems and depth cameras. Stereo vision systems, such as SceneScan [sce, 2018]and ZED [zed, 2018], have been used widely for this purpose, since they can provide dense depth information in short times. Typically, the sensors of stereo vision systems come pre-calibrated, while the underlying technology for computing the depth of the scene relies on stereo algorithms. However, due to their increased complexity, the execution of these algorithms requires a fast host machine, while their poor performance in non-textured and occluded regions makes inevitable the post-processing of the extracted range data. Nevertheless, recent advances in computer vision and acquisition technologies allow stereo-photogrammetric approaches to deliver range data and point clouds at comparable accuracy and density but with superior throughput than laser scanners [Leberl et al., 2010], which makes them beneficial for a wide range of applications.

Depth and time-of-flight (ToF) cameras, such as Kinect and PMD [pmd, 2018] respectively, have been also used for similar applications. Kinect uses structured light to capture the depth of the scene, while it supports reasonable resolution (640×480 pixels) at ranges of approximately 1 to 3.5 meter and at a maximum frame rate of 30 frames per second (fps). Although it provides reliable depth measurements under a large variety of conditions, the acquired depth frames need to be registered with the RGB frames after the acquisition process, while the use

of structured light prohibits the usage of multiple units in the same space due to interference. Considering also its short working range and low resolution, Kinect is more suitable for measuring small indoor areas, rather than large-scale environments (e.g. buildings). PMD and other ToF cameras, on the other hand, acquire the depth information directly on the camera hardware without needing any calibration, but their several limitations, such as their increased cost, the acquisition of only gray-scale images, their limited frame size and the low frame rates restrict their applicability in highly demanding scenarios, such as the 3D reconstruction of building interiors.

B. Reconstruction and Modeling of Indoor Scenes

After measuring and acquiring the raw 3D data of a scene, its three-dimensional model has to be generated. Independently from the acquisition method that will be used, this processing stage is common for all methods and targets to enrich the reconstructed model with additional structural and semantic shape information.

Architectural modeling of building interiors. The problem of modeling buildings interiors is closely related to the more well-studied case of reconstructing the exterior outlines and façades of buildings [Adan and Huber, 2011; Dore and Murphy, 2014], although the challenges to be tackled are different and cannot be easily solved by the techniques used in urban reconstruction approaches. The main reason lies in the inherent difficulties that indoor environments introduce, since they often contain many objects and obstacles which prevent an ideal data acquisition and easy reconstruction of their permanent structural elements.

However, despite the fact that the reconstruction of indoor scenes has been object of active research for many years, existing methods typically rely on very strict assumptions on the environments considered and usually fail to automatically reconstruct their 3D models without human intervention. For instance, many approaches [Sanchez and Zakhor, 2012; Turner and Zakhor, 2012; Oesau et al., 2013; Turner and Zakhor, 2014; Mura et al., 2014] assume that walls are vertical and that ceilings are horizontal, while a number of other methods make even stricter assumptions, requiring that rooms have a convex floor-plan [Ochmann et al., 2014] or that the environments conform to the so-called *Manhattan-world* assumption, i.e. an assumption stating that cities and indoor scenes are built on a Cartesian grid [Okorn et al., 2010; Budroni and Böhm, 2010a]. Furthermore, the majority of them focus mainly on generating a structural building model by reconstructing only the wall surfaces, without focusing on reconstructing higher-level architectural information such as the wall openings, narrowing further down their applicability and the quality of the extracted models. In addition, almost all of these methods impose a hard restriction to their modeling process by relying on

the explicit prior knowledge of the true scanner viewpoint positions, in order their reconstruction pipeline to be successfully executed. Similar limitations are often enforced in the closely related problem of outdoor building reconstruction [Nan et al., 2010; Vanegas et al., 2012], for which however more flexible and expressive methods have been proposed [Tarsha-Kurdi et al., 2007; Chen and Chen, 2008; Chauve et al., 2010; Lafarge and Alliez, 2013]. Furthermore, the inability of many state-of-the-art methods to cope with deficient inputs is another weak point; only few methods [Adan and Huber, 2011; Adan et al., 2013; Mura et al., 2014; Mura et al., 2016; Ochmann et al., 2016] include a robust handling of artifacts and occlusions in their pipeline. More recent approaches have also shifted the attention from simple architectural reconstruction to the detection and extraction of individual rooms [Mura et al., 2014; Turner and Zakhor, 2014; Ochmann et al., 2014; Mura et al., 2016; Ochmann et al., 2016], but despite the noteworthy improvements they offer, they still rely on the assumption of knowing a priori the scanner viewpoint positions for reconstructing the room layouts, while they have also devoted very limited effort to create a fine-detailed building model, where the architectural wall elements would be also reconstructed.

Semantic modeling of architectural wall elements. Beyond the general geometry and shape building interiors, finer details such as the architectural wall elements need also to be detected and modeled in point clouds for the generation of a faithful 3D building model. To achieve this, current methods either make hard-coded assumptions about the geometric properties of the wall elements [Budroni and Böhm, 2010a; Sanchez and Zakhor, 2012; Adan et al., 2013; Boulch et al., 2013] or they rely on proxy models extracted from shape repositories for their retrieval [Funkhouser et al., 2003; Attene et al., 2009]. Moreover, there are approaches that perform the modeling and semantic annotation of these elements after an interactive segmentation phase [Silberman and Fergus, 2011; Shao et al., 2012], whose results however can be assumed to be reliable thanks to the active role of the user. Thus, it can be seen that overall there is still a gap in the current literature with respect to the accurate and reliable modeling of the architectural wall elements of building interiors under real-world conditions.

1.3 Open Challenges

In spite of recent research efforts and the improvements in the acquisition technologies, a fully automatic and generalized reconstruction pipeline, capable to generate faithful and semantically rich 3D models from building interiors has not been presented yet. Current approaches require increased processing times, manual interventions, rely on strict assumptions about the scene or the datasets, and

often underperform under real-world conditions. This is mainly due to the high complexity of the problem, but also because of the many intertwined issues that still remain unsolved and perpetuate or even enlarge the drawbacks and limitations of current state-of-the-art methods.

Although the challenges in the field are diverse and manifold, we highlight below four main open issues that limit the applicability of existing state-of-the-art methods and deteriorate the quality of the generated 3D models. It is worth noticing that for some of them, such as the retrieval of original scanner positions, there is still no solution proposed by the research community until now.

- **3D data acquisition**

  Typically, laser scanning devices can extract directly and with high accuracy the 3D information of the scene, but their limitation to provide information about the surfaces' textures, as well as their high cost, large size – compared to simple handheld cameras, and their very time-consuming data acquisition process limit their applicability to many real-world scenarios and make their usage impractical for time-critical applications. Moreover, the fact that they comprise a closed system which does not allow user interventions to the way they operate, do not allow any space for further improvements during the acquisition and measuring stage. On the other hand, stereo acquisition systems allow the fast reconstruction of the environment through low-cost, portable and high-resolution handheld cameras, but exhibit often increased computational complexity and reduced accuracy compared to laser scanners. Thus, an efficient data acquisition system should be capable to combine increased accuracy and high performance, producing results of adequate quality for the majority of real-life indoor applications.

- **Reconstruction of architectural wall elements**

  Modern buildings are composed by permanent components (e.g. walls), which can include various architectural wall elements, such as windows and doors. Despite the huge variety of shapes and geometric properties they exhibit, these elements should be captured and reconstructed accurately during the modeling process. However, the presence of interior objects and clutter in indoor environments can lead to viewpoint occlusions, which will corrupt their shape, hiding a part of them and making their full reconstruction very difficult. Thus, a valid modeling pipeline should be capable to identify accurately these wall elements and separate them from the occluded regions in the challenging real-world indoor environments, independently from the variabilities they present in shape and size.

- **Semantic annotation of architectural wall elements**

Besides the geometric aspects considered for the automatic detection and reconstruction of the architectural wall elements in indoor environments, a transition to BIM requires also semantic information to be incorporated to the final 3D model. Current methods are capable to semantically segment the building space into disjoint spaces, such as rooms, hallways, etc. but little attention has been given to semantically enrich these permanent structures with information about the wall elements. For this reason, the modeling pipelines should be also capable to semantically segment the wall elements, assigning to them labels according to their properties and functionality.

- **Faithful 3D modeling without assumptions and priors**

  Raw input data from terrestrial laser scanners are often associated with some metadata information (such as the scanning positions), which is typically lost after the point cloud registration process. However, due to its vital importance for the successful reconstruction and modeling of the scene, the majority of modeling pipelines manually retrieve the scanner positions in order to use them as priors for occlusion detection, disjoint space partitioning, structural element detection, wall features segmentation and other. Moreover, the frequent unavailability of point cloud metadata information – and consequently the lack of knowing the original scanner viewpoint positions – can cause additional issues to current reconstruction approaches, such as wrong localizations, imprecisions to object detections and wrong semantic interpretations. Therefore, it is important a valid and automatic indoor modeling pipeline to be independent from such restrictive priors and manual interventions, or to be capable to retrieve on its own any information needed for generating automatically faithful 3D models.

1.4 Research Objectives

The work presented in this thesis focuses predominantly to advance the state-of-the-art in architectural 3D modeling of building interiors by proposing improvements and solutions to the open challenges mentioned in the previous section. More specifically, the research objectives of this thesis can be summarized as follows.

I. Efficient acquisition of 3D data

Terrestrial laser scanners and stereo vision systems are commonly used for capturing the 3D structure of the scene. However, due to the limitations placed on terrestrial laser scanners utility, their usage in a variety of applications is impractical, computationally expensive or even impossible (e.g. in

applications with strict timing constraints or when fast acquisitions are necessary). On the other hand, stereo vision algorithms might yield less accurate depth data than laser scanners, but they are better qualified for time-critical and highly demanding processing applications, such as processing large-scale datasets or multi-store buildings. Therefore, in this work we target to develop an efficient stereo vision algorithm for producing qualitative and dense 3D data under high processing speeds. This method should be also capable to bridge the gap between the highly accurate but slow acquisition systems that rely on laser scanners and the fast but less accurate stereo vision techniques. Such an approach would be beneficial for the data acquisition stage of the 3D reconstruction and modeling pipeline, and it will be more suitable for applications in which timing and processing constraints are critical.

II. Detection of architectural wall elements

Modeling pipelines focus mainly on the successful reconstruction of the permanent building structures, without emphasizing in generating a fine-detailed 3D model. For this reason, one of the goals of this thesis is to improve the modeling capabilities of current state-of-the-art methods, allowing them for a more detailed modeling of the interiors. Emphasis will be given mainly in the detection of the architectural wall elements and the generalization of the method, in order to make it suitable for being embedded as an additional processing unit in the majority of modern modeling pipelines.

III. Semantic annotation of architectural wall elements

A fundamental requirement for BIMs is the incorporation of detailed semantic information to the reconstructed models of the environment. Such high-level semantization should emphasize, not only to the labeling of the individual rooms – as mainly happens by the current methods, but also to the identification and semantic segmentation of their architectural wall elements. For this reason, we aim to enrich the information provided by the reconstructed 3D models, by semantically labeling the architectural wall elements of the environment.

IV. Automatic reconstruction of original scanner positions

Point cloud metadata information and specifically the coordinates of laser scanner positions are used extensively the last years from the majority of 3D reconstruction methods as priors for reconstructing and modeling successfully the interior environments. Hence, all these methods impose a hard restriction to the reconstruction pipeline, limiting their applicability and increasing their dependency to external factors. One of the main objectives of this work is to lift this widely-used restrictive assumption and provide a method capable to

retrieve automatically the scanner viewpoint positions from raw point clouds without requiring any prior knowledge about the dataset or the scene.

1.5 Contributions

This dissertation constitutes the cumulative result of three major contributions framed in a common scenario and context, as depicted in Figure 1.2. This figure outlines the major tasks that constitute a typical reconstruction and modeling process – which is briefly described in Section 1.1 and in more detail in Chapter 2 – and points out the contribution areas of our work.

In a nutshell, the research contributions and the related scientific publications (if applicable) with which we achieve the goals of this thesis can be summarized as follows:

- **High performance stereo vision system for 3D reconstruction**

  In the data acquisition stage, we improve with this work, which is annotated as "1st contribution" in Figure 1.2, the 3D reconstruction process inferred by passive range sensors, introducing a new stereo vision method that fulfills the requirements of real-world applications. The proposed method is capable to produce high-definition (HD) range images in a computationally efficient manner, combining a correlation-based matching cost algorithm and an optimization method based on *Discrete Dynamical Systems* (DDS). In addition, the performance requirements set by this thesis are fulfilled by implementing the proposed approach on a custom high-end field programmable gate array (*FPGA*) device, following a scalable and highly parallel hardware architecture. The approach is described in more detail in Chapter 3 and has appeared as an article in a scientific journal [Michailidis et al., 2014].

- **Automatic reconstruction of architectural wall elements**

  In the modeling stage of the reconstruction process, we propose a method capable to recover and semantically label the architectural wall elements of indoor environments in an automatic fashion (annotated as "2nd contribution" in Figure 1.2). In our approach, the detection and semantic labeling of wall openings relies on a bayesian graph-cut optimization method, which can efficiently segment and classify the openings in wall surfaces under the presence of significant amounts of clutter and occlusion. A competitive advantage of the proposed method is that it allows to be embedded as a post-processing unit to the majority of modern 3D modeling pipelines, enabling their reconstructed models to be reliably and automatically enriched with higher-level semantic information. This contribution, which is described in detail in Chapter 4, has been presented in

a peer-reviewed conference [Michailidis and Pajarola, 2015] and has appeared in an extended version as an article in a scientific journal [Michailidis and Pajarola, 2017].

- **Automatic reconstruction of scanner viewpoint positions**

  In our last contribution (annotated as "3rd contribution" in Figure 1.2), we propose the first method for re-engineering and reconstructing the original scanner viewpoint positions from raw point clouds, lifting the restrictive assumption introduced by the majority of 3D reconstruction methods until now that the true scanner viewpoint positions should be known a priori, in order the reconstruction of 3D BIM models to be performed successfully. Our approach allows the accurate localization of LiDAR devices under very challenging indoor environments by exploiting only information derived from raw data, without requiring any prior knowledge about the environment or the dataset. Moreover, being independent from the laser scanner manufacturers, it can efficiently be applied to merged real-world and large-scale indoor point clouds, where the original scan positions are typically lost and not available. Chapter 5 provides a full description of this contribution, while it has been already submitted for publication in an international peer-reviewed conference.

1.6 Dissertation Overview

This dissertation is structured as follows.

In Chapter 2, we outline the basic aspects of a typical reconstruction pipeline and we provide the proper background, in order to allow a smooth transition to the next chapters, where the details of the contributions of this thesis are provided. Chapter 3 describes a new hardware-based stereo reconstruction method, which is capable to produce high-resolution dense disparity maps in real-time. The same chapter presents also its efficient implementation on a single FPGA device. Chapter 4 describes a robust pipeline for reconstructing the architectural wall elements of indoor environments, while Chapter 5 introduces a novel approach for automatically reconstructing the real scanner viewpoint positions from raw point data. Finally, Chapter 6 concludes this thesis with a summary and gives some promising directions for future work.

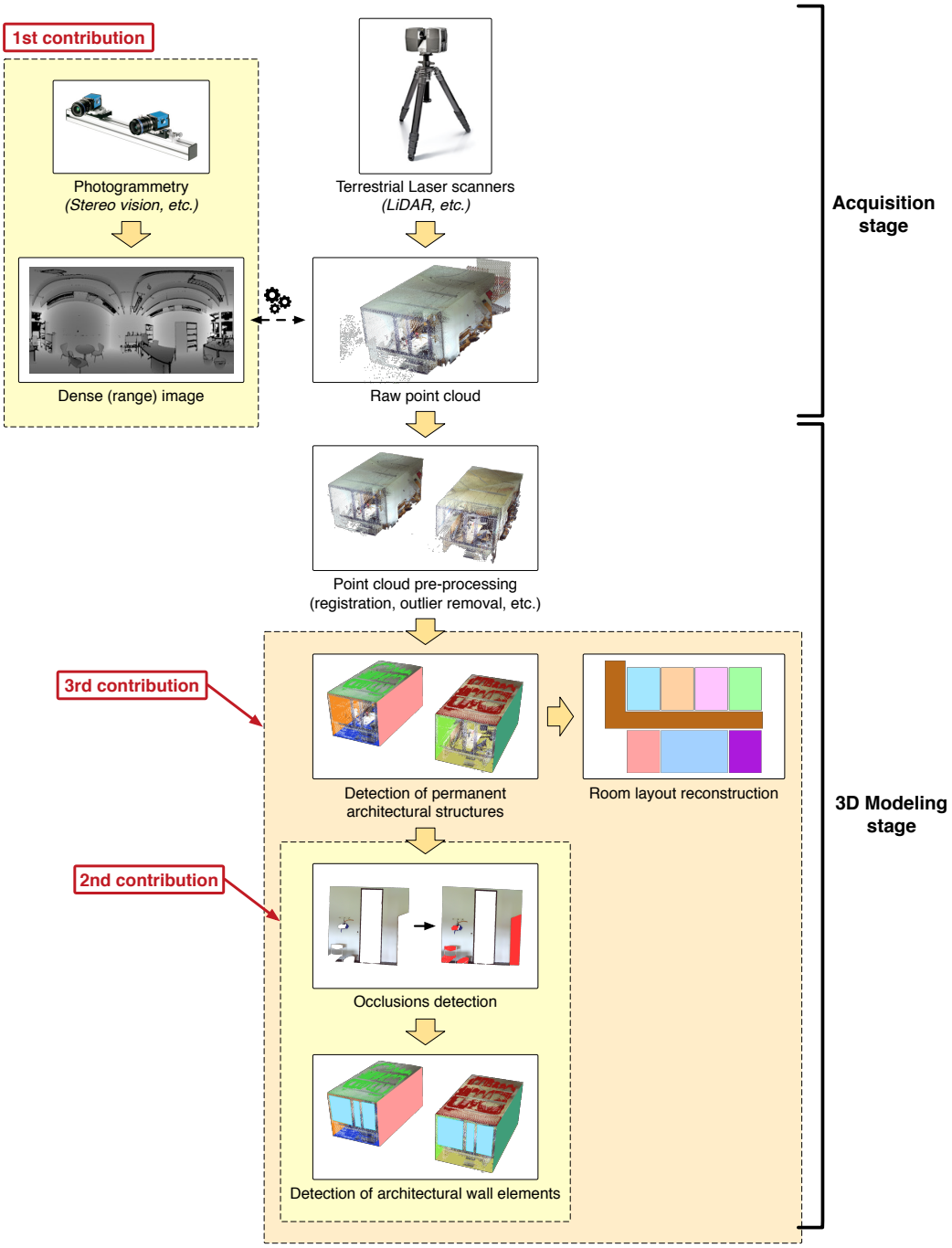
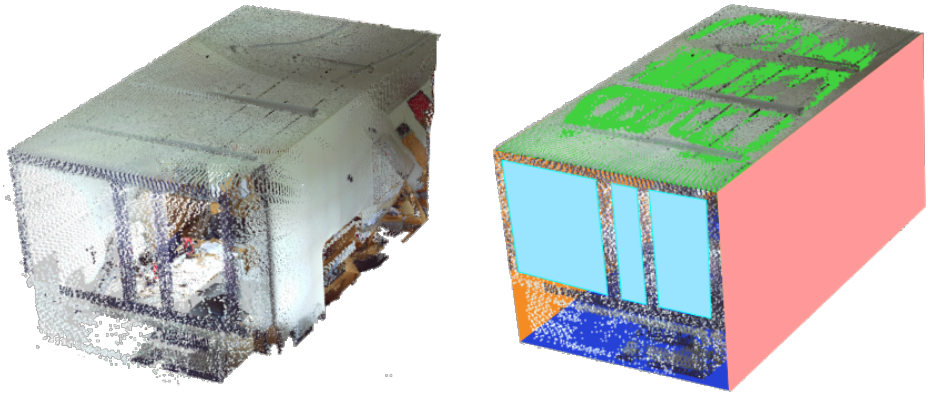


Figure 1.2: *A typical 3D reconstruction pipeline divided into the data acquisition and 3D modeling stage. The annotated parts point out the contribution areas of this thesis (marked with yellow and orange colors).*

C H A P T E R 2

GENERAL 3D RECONSTRUCTION PIPELINE



2.1 Introduction

This chapter discusses the main aspects of a typical reconstruction pipeline and provides a short overview of the main tasks that constitute each of the two reconstruction stages of the modeling process (see also Figure 1.2). Given the vastness of the literature on the topic, focus is placed more on the parts that are related or used by the approaches presented in this thesis.

2.2 Data Measurement and Acquisition Stage

In the first stage of the reconstruction pipeline, the depth information of a scene is acquired, either using passive or active methods [Strecha et al., 2007; Kaller et al., 2016]. Traditionally, there are many different acquisition systems capable of delivering high-quality and dense measurements. The most popular systems that reside in active methods are laser scanners and LiDAR devices, which typically present highly accurate dense measurements, but also high acquisition costs and increased operational complexity. Consumer-level RGB-D cameras and photogrammetric methods such as stereo vision offer a cost-effective alternative for generating the 3D models of interiors, while tablets and low-cost hand-held devices that are equipped with 3D sensors can be turned also into 3D scanning systems [Tanskanen et al., 2013]. Any of these approaches allows the acquisition of the as-is state of an environment, despite their differentiations in the quality of acquired models. Especially for the reconstruction of building interiors, modern reconstruction pipelines use mainly raw data generated by laser scanners or stereo vision systems [Strecha et al., 2008; Leberl et al., 2010], since they present increased accuracy and high-quality results against other acquisition approaches [Zhu and Leow, 2013].

Generally, there are various formats with which the digital model of the environment could be represented. The most precise 3D representation and often the direct output of laser scanners is the 3D point cloud, which is usually used by higher-level 3D modeling pipelines as an initial raw representation of the digital model of the scene. Another format for representing the acquired 3D space – and usually extracted by passive methods – is the range image (or depth map), which contains precise information about the distances to the points in the scene and can be also used to generate the equivalent 3D point cloud by just applying a simple transformation algorithm.

In the following sections, we provide more information about the fundamental concepts of laser scanners and stereo vision, as the contributions discussed in the next chapters rely on them.

2.2.1 Terrestrial Laser Scanners

Laser scanners (LS) are electronic distance measurement (EDM) devices that can be used in various applications for measuring the distance between two points by emitting an electromagnetic radiation (a laser light beam). Based on the application and their usage, they can be placed on different platforms such as airborne (ALS), terrestrial (TLS), mobile (MLS) or satellite.

In general, laser scanners can determine the measuring distance to the object of interest relying on either Phase-Shift or Time-of-Flight (ToF) sensors. Regardless from the technology used, the laser beam or pulse bounces back from the target object surface and returns to the instrument, where either the phase difference between the emitted and the reference beam is measured, or the time for the round-trip of the beam is determined, respectively.

To scan the surrounding area, laser scanners use a rotational mechanism coupled with a rotating polygon mirror, in order to deflect the transmitted laser beam to different directions and cover the area around them. The coverage of the surrounding area is also dependent on the *field-of-view* (FOV) of the laser scanner, which corresponds to the observable area that the laser scanner can capture from a specific position. Most commercial laser scanners produced by the main manufactures (e.g. Leica, Mensi, Riegl, OpTech, Faro, etc.) present usually a FOV of almost $360°$, forming a closed spherical scan region around their scan position and excluding only a small part beneath them due to the presence of the supporting tripod.

Figure 2.1 illustrates a typical scanning setup of such a LiDAR device. During

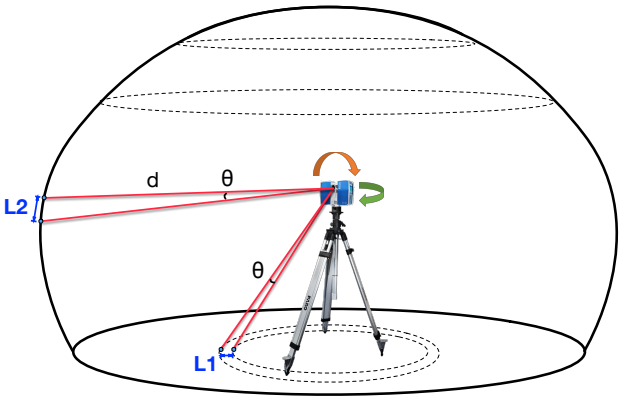


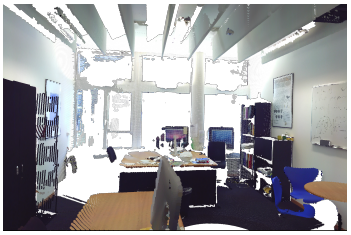
Figure 2.1: *Typical laser scanning setup, showing the movement of the rotating mirror around the horizontal and vertical axis, as well as the angle θ of the angular scanning increments. Notice that the density of points near the scanner position is higher than in areas farther away (i.e. $L1 < L2$).*

the scanning process, the laser device scans the FOV point-by-point by slewing the laser beam around using the rotating mirror. The return energy of the emitted beam is then caught by the receiver of the laser sensor, detecting the strength of the backscatter (i.e. the so-called "intensity"), the distance and the reflective characteristics of the object surface [Heritage and Large, 2009]. These measurements can then easily be transformed into a set of 3D points, expressed by the triplet of point coordinates and the other point attributes (e.g. color or distance). In addition, the same measurements can be also transformed into a range (depth) image, in which the Euclidean coordinates of the measured 3D points are converted into spherical coordinates expressed by two angles and the distance d between the scanner and any point in space (see distance d in Figure 2.1). In this case, the range map of the scene is generated by storing these distances to an image [Swadzba et al., 2007; Herbort and Wöhler, 2011]. One such an example is presented in Figure 2.2.

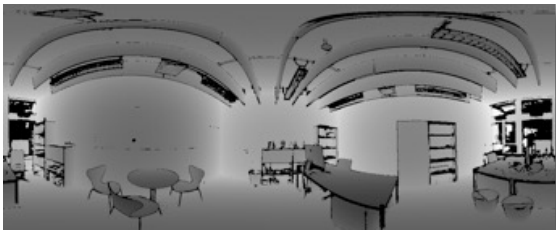
Notice here however that, due to the spherical movement of the rotating mirror, the scenes acquired by laser scanners exhibit regularly a *circular* scanning pattern, which can be easily detected in areas mainly near the scanner positions. Figure 2.1 illustrates one such an example, where the acquired points near the scanner standpoint form two concentric circles centered at the scanner position.

Moreover, due to the nature of the scanner acquisition process, there are some

(a) Panoramic 360° image from room interiors.



(b) 3D point cloud.



(c) Range (depth) image.

Figure 2.2: *The depth values obtained during the acquisition process of a room interiors shown in (a), can be used to produce a range image (b) or to form a 3D point cloud (c).*

important factors that determine the intrinsic scanning characteristics and consequently the quality of the generated point-based digital models. For instance, the user selected (angular) resolution of the device – commonly known as scan density – specifies the smallest possible increment of the angle between two successively measured points, while it remains constant during the whole acquisition process (see angle θ in Figure 2.1). Therefore, the density of measured points is relative to the distance from the scanner positions, meaning that the scanned surface areas near the scanner positions present higher density of samples (points) than the areas further away. In Figure 2.1, this is illustrated by the distances $L1$ and $L2$, where keeping the angle θ constant, the distance between two consecutive points near the scanner position ($L1$) is smaller than the distance between two points farther away ($L2$).

2.2.2 Stereo Vision

Stereoscopic (in short, stereo) vision systems have been used extensively as substitutes of 3D laser scanners in applications where their usage is not possible, feasible or cost-effective, such as in automotive industry, in specific industrial and time-critical applications, in indoor environments, etc. [Strecha et al., 2008; Zhu and Leow, 2013]. Due to the rapid evolution in digital cameras, stereo reconstructed 3D models can provide detailed structure information about buildings and interior environments, generating 3D point clouds which present very little difference with the LiDAR point clouds [Hirschmuller, 2008; Leberl et al., 2010; Agarwal et al., 2011; Ok et al., 2012; Steinhage et al., 2013].

The concept behind the acquisition of depth information from stereo vision systems is quite simple. Two (or more) stereo images are taken from slightly different viewpoint positions, and the three-dimensional structure of the world is inferred by comparing the information derived from these images [Scharstein and Szeliski, 2002]. Specifically, and similar to human vision, the overlap occurred between the stereo images is used to perceive the depth of the scene by computing the disparities, i.e. the differences in the placement of objects as viewed by the different viewpoints. The fundamental geometric concept of a binocular stereo setup is shown in Figure 2.3. In this ideal system, the stereo cameras are horizontally aligned and separated by a distance known as baseline, in order to capture the two (stereo) images. The optical axes O_L and O_R of the two cameras are perfectly parallel and both stereo image planes are coplanar, assuming that no lens distortion is present. To reconstruct the 3D model of the scene, the distance Z to point P has to be determined using the following equation:

$$Z = \frac{B \cdot f}{d}, \tag{2.1}$$

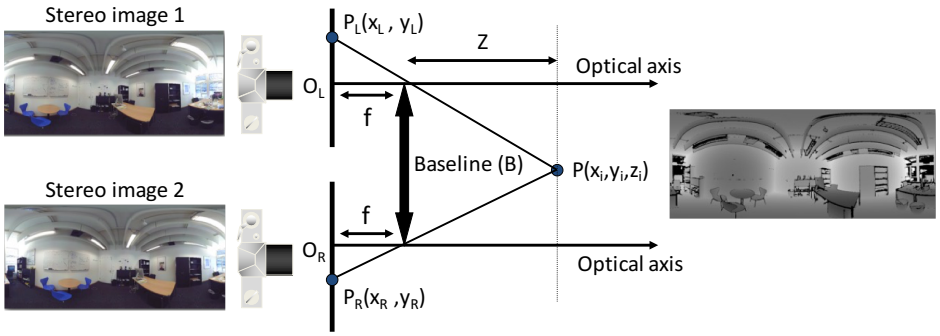


Figure 2.3: *Ideal geometry of a stereo setup.*

where f is the focal length, B the baseline and d the disparity. The baseline describes a common line for the different views of the scene and constitutes the basis for finding the correspondences between the stereo images. Applying in this ideal scenario a simple similarity metric for comparing directly the pixel values, the disparity of P can be defined as

$$d = x_L - x_R. \tag{2.2}$$

The computation of distance Z for all objects in the scene will allow for the creation of its range image (or disparity map), while the range data produced by the stereo vision system could be used to generate the equivalent 3D point cloud of the scene, as depicted for example in Figures 2.2(b) and 2.2(c).

To apply, however, Equation 2.2 for computing the disparity map, we need first to overcome some restrictions introduced to the disparity computation process. Finding and matching the corresponding points P_L and P_R between the stereo images, i.e. solving the stereo correspondence problem [Scharstein and Szeliski, 2002], requires the points to lie on the same image line, called the epipolar line. Otherwise the complexity of the stereo matching problem increases dramatically. However, in typical real-world scenarios the epipolar lines might be slightly slanted, since the cameras could be moved towards the object or the raw stereo images could be distorted by the camera lenses. Applying a process called rectification [Lim et al., 2004] (see Figure 2.4), the images can be warped in order to appear as if they have been taken with only a horizontal displacement, and hence, all epipolar lines to become horizontal. Then, the computation of disparities becomes feasible based on Equation 2.2.

Although Equation 2.2 presents a simple metric for computing the disparities inside a predefined disparity range, due to the noise and other distortions introduced to the input stereo images, more robust methods are typically applied for

(a) Slanted epipolar lines due to image distortions.

(b) Removing image distortions make epipolar lines straight.

(c) Rectification aligns epipolar lines with image axes.

Figure 2.4: *After rectifying the stereo images, the correspondence search can be performed along the image scanlines.*

the disparity computation. Traditionally, these methods are categorized in the literature into global and local methods, and a good survey on the subject can be found in [Scharstein and Szeliski, 2002; Brown et al., 2003]. Global methods are iterative algorithms that rely on minimizing the energy of a global cost function and are capable of generating quite accurate depth data at the cost of high computational complexity and low processing speeds. Local methods are faster but less accurate, and based usually on correlation algorithms they attempt to determine the depth of the scene. The most commonly used global methods try to solve the stereo matching problem using Dynamic Programming (DP), Belief Propagation (BP) and Graph Cuts (GC) [Veksler, 2003; Gong and Yang, 2005; Veksler, 2005], while local methods attempt to address the correspondence problem using similarity metrics based on block matching, gradient matching and feature matching algorithms [Psarakis and Evangelidis, 2005; Nalpantidis et al., 2008; Sharma et al., 2011; Pham and Jeon, 2013].

2.3 3D Modeling Stage

In this stage of the reconstruction pipeline, the goal is to transform the raw and unstructured sets of measurements into a compact and semantically rich archi-

tectural 3D model. Due to the complexity of the problem, usually the 3D point cloud format is preferred over the depth images as the raw input representation of the environment in this stage, since it allows algorithms to be more flexible and exploit more information from a three-dimensional space.

In the next sections, the most important and commonly used processing steps for modeling the building interiors are briefly discussed, as illustrated also in Figure 1.2.

2.3.1 Point Cloud Pre-processing

The first step for reconstructing the model of the scene is to assure its faithful representation. Including a sufficient number of samples in the source point cloud usually helps in this direction, since it reduces the sparse and unexpressive regions in raw data while eliminating the occluded areas. Thus, the scene of interest is often scanned multiple times from different positions, depending on its complexity and the level of occlusions occurred by the interfered objects.

Since the samples of each scan are expressed in a local reference system, to create a single point cloud from multiple scans it is necessary to align the acquired scans into a common reference frame. This is achieved by associating each scan with a matrix, which expresses the rigid body transformation that has to be applied to the points of each scan, in order to transfer them from their local reference system to a common global one. This process is called *registration* and can be performed either manually or automatic [Akca, 2003; Von Hansen, 2006; Boehm and Becker, 2007; Schenk and Hanke, 2012].

After registration, the metadata information (e.g. the scanner positions) from each individual scan gets lost and only point-related data (e.g. point coordinates, color and depth) remain in the registered point cloud. Since in many cases the raw point clouds are affected by clutter, outliers, noise and large-scale artifacts, it is common practice to apply a pre-processing filtering process to the merged point cloud in order to eliminate (partially) these discrepancies and simplify the operation of the modeling algorithms.

2.3.2 Detection of Permanent Architectural Structures

After registration, the merged point cloud typically contains millions of points that make difficult and inefficient the direct application of a modeling algorithm on them. Therefore, depending on the application and targets set, this point data representation can be simplified by utilizing first a clustering method, which will group the points with similar properties into patches, reducing the immense number of data considered in the following stages of the modeling process. Various clustering methods have been used successfully so far for this occasion, such as

K-means clustering [Shi et al., 2011], hierarchical clustering [Feng et al., 2014] and fuzzy c-means clustering [Wang et al., 2006], to name a few. Next, the point patches are merged to form larger segments, or they are replaced by their centroids, which will be used as patch-representative 3D points in the next processing steps.

Besides clustering, it is also very common in the first stages of the modeling process, geometrically simple parts of the scene to be detected and represented by shape proxies. Methods such as Hough Transform [Vosselman et al., 2004], least squares [Gruen and Akca, 2005], region growing [Poppinga et al., 2008] and RANSAC [Schnabel et al., 2007] have been used extensively for this purpose, extracting simple geometric shapes from point clouds. Especially for the reconstruction of building interiors, this low level shape extraction process is widely used for detecting planar regions, since these man-made indoor environments are mostly composed by piecewise planar surfaces [Steadman, 2006].

Once a point cloud has been segmented into (planar) patches, their geometric and spatial features are exploited, in order to assess their semantic content and investigate their potential classification into specific building element classes such as ceiling, floor, and walls. The classification of the patches is usually achieved either by supervised approaches, where the semantic categories are learned from a training dataset of annotated data, or by unsupervised approaches, where the data is automatically partitioned into segments based on a user-provided parameterization. Common classification methods that lie in these categories rely mainly on machine learning [Zhan and Liang, 2011], Gaussian Mixture Models [Lalonde et al., 2006] and hierarchical K-means classification [Chehata et al., 2008].

Once the patches get classified, the patches that do not belong to the predefined classes are excluded from the modeling process and only the patches that comprise the structural elements of the building [Macher et al., 2017] are kept for further processing.

2.3.3 Floor Plan and Room-Layout Extraction

After patch classification, the majority of modeling pipelines try to address the problem of detecting the sub-spaces in building interiors. This includes the detection of the rooms that compose the environment and the extraction of their shape. Current attempts that semantically classify the rooms either rely on their geometric proportions [Mozos et al., 2005; Sousa et al., 2007] or on a visibility analysis performed between the classified patches and the scanner viewpoint positions [Xuehan et al., 2013; Mura et al., 2014; Mura et al., 2016; Ochmann et al., 2016; Ambrus et al., 2017]. Note that the later methods, in order to reconstruct successfully the general room layout of the building, rely exclusively on the prior knowledge of scanner viewpoint positions, an assumption that does

not always hold for the majority of real-world scenarios, where multiple scans get merged (i.e. registered) into one point cloud and this information is not available anymore.

It is also worth mentioning that, the majority of recent pipelines do not proceed further than this modeling stage, extracting only the shape and layout of the rooms (i.e. the architectural building model) and omitting the detection and reconstruction of the finer room details. Nevertheless, the 3D BIM models need to be accompanied by higher-level information, such as the wall openings, thus these architectural building models could serve as the basis for the generation of higher-level BIM models.

2.3.4 Architectural Wall Elements Detection

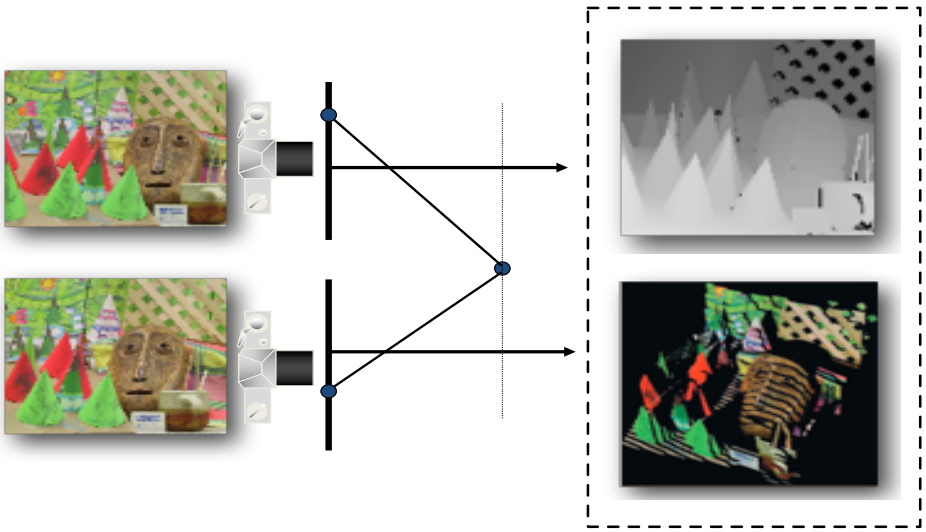
The detection and modeling of the architectural wall elements is a complex and demanding task, due to the inherent difficulties introduced to their reconstruction process. Initially, the occluded areas in the wall surfaces need to be detected and recovered, in order the wall openings to be reliably reconstructed in the next stage. Despite the fact that during the previous reconstruction steps an occlusion detection and handling mechanism was not very important, for the wall openings' reconstruction this process is of vital importance. Current modeling pipelines use various occlusion detection mechanisms for this purpose, either carrying out a visibility analysis on the wall surfaces from the scanner viewpoint positions, or relying on additional clues from imagery and external data.

After the identification of the occluded regions, some reconstruction approaches analyze the point densities in the wall surfaces in order to detect the gaps that correspond to the wall openings, or they calculate the intersections of the previously merged planar patches, in order to identify the edges of the wall openings [Hinneburg and Keim, 1998; Dumitru et al., 2013]. Alternative techniques, relying on spatial and functional relationships between nearby structures and spaces, have been also proposed for reconstructing the architectural wall openings [Adan and Huber, 2011; Xuehan et al., 2013].

Eventually, after their successful reconstruction, all detected wall elements are integrated into the architectural building model and a higher-level representation of the environment, closer to a 3D BIM model, is generated.

C H A P T E R 3

STEREO VISION FOR 3D RECONSTRUCTION



3.1 Introduction

In this chapter, the first research contribution of this thesis is described, which allows the computation of high-definition (HD) range images in real-time or near real-time by an efficient hardware-based stereo vision system. The proposed method relies on the general principles of stereo vision described in Section 2.2.2 and uses an optimized stereo matching technique for finding the correspondences between the input images. In order to achieve higher processing speeds and improve the computational performance, the whole range map computation pipeline was implemented on a custom hardware device, making this method suitable for many indoor applications where LiDAR devices cannot be used for acquiring fast and cost-effectively the depth information of the environment.

Based on the general reconstruction pipeline outlined in Chapter 2, the work presented in this chapter contributes to the annotated parts of Figure 1.2 that refer to the first contribution.

3.1.1 Motivation

Stereo vision is a cost-effective, fast and widely used technique for acquiring the 3D information of the scene, targeting mainly at close-range applications such as the modeling of interior scenes and indoor environments. Although there are other methods that can capture the scene with higher accuracy (e.g. LiDAR scanners), due to its high performance and ability to be executed in fast processing speeds, stereo vision offers some competitive advantages that makes it suitable over a range of scenarios and applications, such as in robotics, navigation, object recognition and surveillance, especially when time constraints are involved. In such a scenario, it is more important to reconstruct adequately and fast the scene structure, rather than to have a highly accurate but slow reconstruction of it.

Various stereo vision methods have been proposed so far for computing the depth information of the scene (see also Sections 1.2 and 2.2.2), which are classified in the literature into global and local methods [Scharstein and Szeliski, 2002]. Global methods are very accurate and can result to dense depth information comparable to that of laser scanners, but they are computationally more expensive and time-consuming. Moreover, they often exhibit irregular data access patterns and are usually unsuitable for time critical applications. On the other hand, local algorithms yield less accurate disparity maps due to their poor performance on textureless and occluded regions, but they are better qualified for time-critical and highly demanding applications, such as processing large-scale datasets.

Therefore, an efficient stereo vision method, capable to perform fast, avoiding the global metrics and improving the quality of the resulting range images, would be beneficial for many real-world applications. Implementing additionally such a

method on a custom hardware device, it would increase significantly its computational performance, allowing the extraction of higher resolution range images in shorter processing times.

3.1.2 State of Research

While a large number of stereo vision methods have been developed, the majority of them had focused on software-based techniques implemented in host computers. Despite the flexibility of these methods and the convenience with which they can be implemented, they present some major limitations, which restrict them to handle the outlined application requirements efficiently. The most restrictive limitation lies in their serial operation, which makes necessary the use of high-end computers or sophisticated code optimization techniques for their execution, increasing drastically their computational complexity and execution time.

The last few years, recent advances in custom hardware devices have allowed researchers to focus more on implementing stereo vision methods directly on *field programmable gate arrays* (FPGAs) or *application specific integrated circuits* (ASICs), since these devices offer high processing speeds and adequate hardware resources for executing efficiently the stereo algorithms. As these methods are more relevant to the proposed hardware-oriented stereo reconstruction approach, we will focus in this section our review only on these hardware-based methods.

An efficient stereo vision system, called *DeepSea*, is introduced in [Woodfill et al., 2004], which relies on a specialized ASIC processor in order to compute disparity maps of size 512×480 at 200 frames per second (fps) and up to 52 different disparity levels. A different stereo vision system, which relies on a FPGA architecture to generate disparity maps of 512×480 stereo images at 30fps is presented in [Hile and Zheng, 2004], where a window-based and scan-line stereo matching algorithm is used for finding the corresponding points and computing the disparity values.

Authors in [Miyajima and Maruyama, 2003] present the hardware implementation of a stereo method, which is capable to produce disparity maps at 20fps using VGA images. However, the performance of this system is quite limited, mainly due to the utilized memory access pattern, which does not allow for scalability and performance optimization. A miniature stereo vision system relying on trinocular stereopsis is presented in [Jia et al., 2004], which uses a Xilinx FPGA platform for performing all stereo-related tasks, such as trinocular rectification, LoG (Laplacian of Gaussian) filtering and stereo matching. This system can run at approximately 30fps with 640×480 pixel images within a disparity search range of 64 pixels.

Other stereo vision methods, which focus mainly on local SAD-based techniques for computing the depth images, are proposed in [Lee et al., 2005; Hariyama

et al., 2005; Ambrosch et al., 2009; Gudis et al., 2012]. These methods can reach near real-time processing speeds and are capable to produce range images of various sizes (up to 640×480) with disparity levels up to 64. The stereo method presented in [Lu et al., 2007] uses directional center-biased support windows for computing the matching cost values, utilizing an efficient GPU-based optimization scheme that can extract range images with sizes up to 512×512 and disparity levels up to 96 in near real-time. The work described in [Sabihuddin et al., 2008] implements a FPGA-based method that can generate dense disparity maps at high frame rates (up to 200fps) using a dynamic programming maximum likelihood (DPML) formulation, while in [Banz et al., 2010], a scalable systolic-array-based architecture was used for extracting VGA depth images with 128 different disparity levels under real-time conditions (30fps).

There are also methods [Šára, 2002; Hirschmuller, 2001; Giryes et al., 2008] that utilize specialized hardware such as Intel MMX processors, graphics processing units (GPUs) and digital signal processors (DSPs) for reconstructing the 3D scene structure. Although these methods can solve the 3D reconstruction problem in a computationally efficient manner, they are not suitable for embedded and mobile applications due to their increased power consumption and high resource utilization. Also, some of them target mainly on face recognition applications rather than the reconstruction of spatial environments.

3.1.3 Research Contribution

The contribution discussed in this chapter attempts to diversify from the state-of-the-art in the field and follows a new path for refining the generated range images. Specifically, we present a novel approach that accesses the intermediate level of the disparity estimation process and refines the matching cost values in the 3D *disparity space image* (DSI) domain, as opposed to the current stereo reconstruction methods, which intervene in the pre- or post-processing steps for improving the disparity maps. Our method is applied *before* the disparity selection procedure and after the application of the similarity metric, which allows to substantially refine the computed depth values, resulting in range images with better quality than the related state-of-the-art methods in shorter time. We also present in this chapter the implementation of the proposed stereo reconstruction approach on a custom high-end FPGA device, following a scalable and highly parallel-pipelined hardware architecture.

Although our method could be classified as local, its ability to exploit more information and analyze data that lie in 3 dimensions and not only in 2, as it happens in pre- or post-processing steps of other techniques, constitutes its big advantage. Another beneficial feature of our approach is its capability to improve the disparity maps by optimizing only the selection process of the pixel disparity values

without using any additional processing steps, while being independent from the similarity accumulators makes it suitable for embedding it as an intermediate optimization stage to the majority of local-based stereo algorithms. Finally, it is worth noticing that the extracted range images are produced without applying any kind of post-processing, meaning that any post-processing step could further improve the extracted disparity maps.

3.2 Method Overview

Following the general stereo reconstruction pipeline described in Section 2.2.2, our method takes as input two stereo images and process them in order to compute the depth of the scene. A brief overview of the proposed stereo reconstruction pipeline, which is visually summarized in Figure 3.1, is shortly introduced in this section, while an extensive description is given in the next sections of this chapter.

In general, the contribution of this work could be separated into two parts, the proposed stereo reconstruction method that computes the range images and the hardware architecture that implements the proposed method.

Stereo-based 3D scene reconstruction In the proposed approach, the input stereo images are first pre-processed for smoothening the noisy regions and preserving the intensity consistency among the pixels. Then, the matching costs between the pixels of the input images are computed by a window-based correlation algorithm, and the resulted accumulated values form the initial 3D space of DSI (see also Figure 3.1). In this three-dimensional domain, we apply an efficient optimization technique, which relies on Discrete Dynamical Systems and Cellular Automata, in order to refine the selection process of the best matching costs and improve the effectiveness of the similarity accumulator. In the final stage, a similarity accumulator selects the best matching score for each pixel and assigns to it the proper disparity value. The output range image is finally generated by the reconstructed depth values, while based on the application requirements, it could be further converted into a 3D point cloud, in order to be used as input to the modeling stage of the reconstruction pipeline, as it is indicated in Figure 1.2.

Hardware Architecture The implemented hardware architecture of our stereo reconstruction approach includes three main units responsible for pre-processing the stereo images, computing the accumulated matching costs and refining the DSI. The proposed system design includes also a dedicated processing unit for coordinating the operations performed in the architecture and for optimizing the memory accesses. It allows also the user to set specific configuration parameters that influence the performance of the reconstruction process, such as

the maximum range of the disparity values and the window size of the correlation algorithm.

Before proceeding to the detailed description of the proposed stereo reconstruction method, we find particularly useful to provide first a short overview of the Discrete Dynamical Systems and Cellular Automata.

3.3 Overview of Discrete Dynamical Systems and Cellular Automata

Discrete Dynamical Systems (DDS) are systems whose behavior changes dynamically over time, while their state is known only at a discrete set of times. Using a mathematical formalization, a DDS consists of an abstract phase space S, which is also called as state space. The coordinates of S describe the state at a set of times T, while a dynamical rule R defines the evolution of states according to $R : S \times T \rightarrow S$, from which the future values of all state variables can be specified based on their present values. The evolution of a DDS could be based either on a deterministic model, where for every state there is a unique consequence, or on a stochastic or random model, where the state consequences of the system rely on a probability distribution function [Meiss, 2007].

Cellular Automata (CA) is a core topic in the science of complexity and have been used widely in many application fields such as in image processing, surveillance, robotics and other. CA are dynamical systems introduced by Ulam and Von Neumann [Von Neumann, 1951], which present improved capabilities in massive parallelism and can model complicated systems or perform complex computations with the help of only local information.

Computational algorithms relying on CA present similar features as the DDS systems, i.e. they are discrete in space and time and operate on a lattice of sites. The lattice of sites could be imagined as a grid of cells, and a typical example of such a lattice is an image, where each pixel comprises a cell of the grid. As such, CA consist of an array of cells, where each cell can be in one of a finite number of possible states. The states of a CA structure are updated synchronously in discrete time steps (clock cycles) and in each time step, a local transition rule (cell rule) is applied to the CA grid in order to define the next states of the cells. Usually, the state of a cell at the next time step is determined by the application of the transition rule to the neighboring cells of the current cell, and based on the outcome of the applied rule, the state of the current cell is updated.

Generally, a CA is characterized by the number of spatial dimensions and a triple $\mathcal{A}=(S, N, \mathcal{F})$. The state set S determines the states of the cells, while the set $N \subseteq \mathbb{Z}^2$ defines the neighborhood of each cell to which the transition rule will be applied. Finally, the set $\mathcal{F}$:$S^N \rightarrow S$ defines the local transition function (rule)

which will be applied to the grid cells and will determine the states of cells at time $(t+1)$, assuming that it was applied at a given time (t).

3.4 Stereo-Based 3D Scene Reconstruction

3.4.1 Input Stereo Images Pre-Processing

A common difficulty faced by the majority of stereo algorithms is the identification of the corresponding points. The introduced noise from sensor devices and the illumination differences of objects surfaces distort the pixel intensity values, causing erroneous matches during the matching process. In addition, the textureless regions, the depth discontinuities near object boundaries and the occluded areas make the stereo correspondence problem even harder.

To reduce the inconsistencies in pixel intensities and subsequently the wrong pixel matches in the stereo correspondence stage, we initially pre-process the stereo images by applying a weighted mean filter. Assuming that the input stereo images are already rectified, the filter we used can be described by the following equation:

$$I'(x,y) = \frac{1}{4}(I(x-1,y) + I(x+1,y)) + \frac{1}{2}I(x,y) \tag{3.1}$$

where I is the original and I' the filtered image.

Notice that the application of a two-dimensional filter could produce better results, but the usage of such a filter may not justify the increase in the computational cost that this entails, considering the additional hardware resources and the significant overhead in processing time.

3.4.2 Disparity Space Image Calculation

The next step for generating the depth map of the scene is to solve the stereo correspondence problem and calculate the disparity values. This is the most important and time consuming stage in the stereo reconstruction pipeline, and is divided into three separate tasks, as it is shown in Figure 3.1.

Accumulated matching cost computation To determine which pixels on the left and right images map to the same point in space, we need to compute the matching costs for all disparities under consideration. Taking into account the speed issues and considering the hardware complexity, in our method we used the Sum of Absolute Differences (SAD) of intensities as a similarity measure, which is quite robust to local intensity variations and can be efficiently implemented in hardware. The calculation of SAD is straightforward and for the rectified input

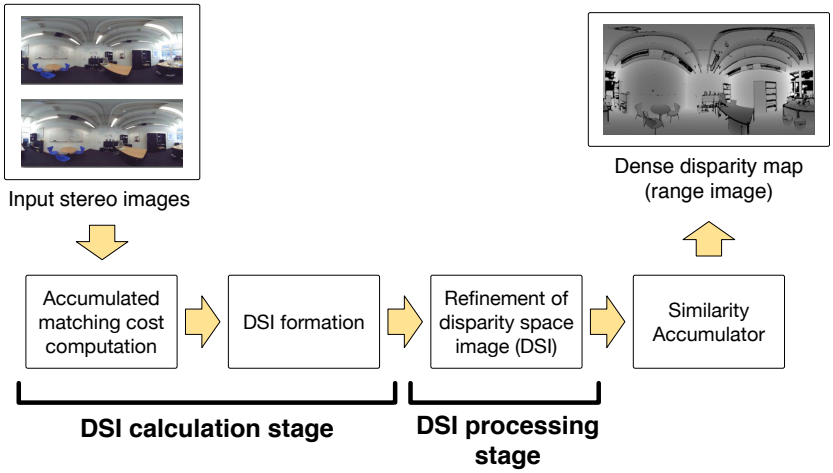


Figure 3.1: *General overview of the proposed stereo reconstruction system.*

images can be computed by the following equation

$$SAD(i,j,d) = \sum_{\mu=-r}^{r} \sum_{v=-r}^{r} |I_L(i+\mu, j+v) - I_R(i+\mu, j+v+d)|, \qquad (3.2)$$

where d refers to a disparity value inside a predefined disparity range, r is the radius that determines the window size around the central pixel and i, j are the center pixel coordinates (rows, columns) of the window for which the SAD is computed. In our implementation, the window size was set by default to 5×5 pixels (i.e. r was set to 2), although its width w constitutes an input variable which can be set by the user according to the application requirements (see also Table 3.7 about how w affects the performance of the system). Notice also that the input image pairs used by our system are RGB color images, which means that the SAD method needs to be applied to each color component separately.

DSI Formation After calculating the SAD for all pixels in the reference image, the best disparity value for each pixel has to be determined. The set of accumulated matching cost values computed by the SAD form a 3D cost volume, often called disparity space image (DSI), which is defined as a 3D space of (x, y, d), where the first two coordinates represent the dimensions of the input images with size $W \times H$, while the third one refers to the matching cost for each disparity value d that belongs to a predefined disparity range $d_{\max}$ (see also Figure 3.2) [Yang et al., 1993]. Given the stereo image pair, the values of DSI can be

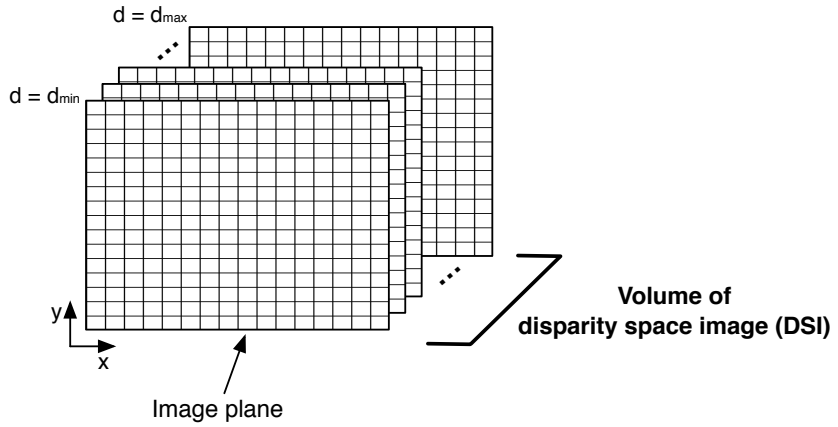


Figure 3.2: *A disparity space image (DSI) representation, where each grid cell value corresponds to a matching cost value calculated from the similarity metric (SAD).*

computed by:

$$\begin{aligned} DSI(x,y,d) &= |P_L(x,y) - P_R(x,y{+}d)| \\ where\ \ 0 \leq (x{+}d) \leq W\ \ and\ \ d_{min} &\leq d \leq d_{\max}. \end{aligned} \tag{3.3}$$

Each element of DSI is a confidence measure and represents the cost (likelihood) of the correspondence between $P_L(x,y)$ and $P_R(x,y{+}d)$ on the same epipolar line, where P_L and P_R are the aggregated matching cost values of the same epipolar line calculated by the SAD, i.e.

$$P(x,y) = \sum_{\mu=-r}^{r} \sum_{v=-r}^{r} I(x+\mu, y+v) \tag{3.4}$$

3.4.3 Disparity Space Image Processing

In traditional stereo reconstruction methods, after computing the accumulated matching costs, the best disparity value for each pixel in the reference image is selected using a winner-takes-all (WTA) selection procedure and the extracted disparity values are then post-processed, in order for the wrong disparities to be eliminated.

However, although the SAD algorithm is quite robust and simple, it does not exhibit high accuracy and it introduces several mismatches that clutter the resulted range images. Inherent ambiguities in occluded areas, regions with periodical structure or unstructured regions, produce random incorrect matching costs, which can be located anywhere in the DSI. As a result, the effectiveness of the similarity

accumulator is reduced and many random incorrect disparity values are introduced in the extracted disparity map.

Refinement of DSI In order to enhance the range images with the minimum loss of 3D information, an efficient CA structure was applied to DSI. Our novel CA approach aims to refine the DSI and improve the efficiency of the similarity accumulator, producing dense disparity maps with minimal false reconstructions.

Since there are numerous rules that can be applied to improve the quality of DSI, a considerable effort was devoted to explore the effects of different rules under various indoor environments and use only those that have proven to be good in eliminating the wrong matching costs in the DSI.

The cellular automaton used in the proposed system can be formulated by the tuple (I, V, N, f), where $V{=}0, 1, \ldots, k-1$ is a set of cellular states and k the number of possible elementary states. N represents the type of neighborhood for each transition rule, f denotes the local transition rule, and I is the cellular space formed by a 2D array of $W \times H$ cells as $I = (a, b)\ s.t.\ 1 \leq a \leq W, 1 \leq b \leq H$, where W and H are the dimensions of the disparity image. The value $DSI^{(t)}(i, j, d)$ of a site (i, j, d), where d is the disparity value and $DSI^{(t)}$ the matching cost value in DSI at time step t, is a 3-dimensional cellular automata with a rule that depends only on nearest neighbors and evolves according to equation:

$$DSI^{(t+1)}(i, j, d) = f[h^t(a)], \tag{3.5}$$

where $h^t(a) = (DSI^{(t)}(\alpha{+}\delta_1), \ldots, DSI^{(t)}(\alpha{+}\delta_n))$ is the neighborhood state function of cell α, for all $\alpha \in W \times H \times d$ and $\delta_{i=1,2,\ldots,n} \in N \rightarrow W \times H \times d$ at time t.

Thus, based on the CA structure we previously defined, we considered the following transition rules for refining the DSI.

CA Rule 1 For each disparity level d and for each element in the same disparity level, set the values of DSI in the next time step as:

$$DSI^{(t+1)}(i, j, d) = \frac{1}{9} \sum_{m=-1}^{1} \sum_{n=-1}^{1} DSI^{(t)}(i{+}m, j{+}n, d)$$

where N is the Moore neighborhood of a 3×3 pixel mask. Since pixelwise cost calculations are generally ambiguous and wrong matches can easily have a lower or higher cost than the correct one, with this rule we eliminate the amount of cost variations that are unrepresentative of their surroundings in a small neighborhood of the same disparity plane and smooth the costs in uniform areas. These variations may be caused by different factors, such as the noise in input images or the illumination differences in object surfaces.

CA Rule 2 For each element of DSI with same (i, j) coordinates and for different disparity levels, set the values of DSI in the next time step as:

---

if $DSI^{(t)}(i,j,d{-}1) > \frac{1}{2}DSI^{(t)}(i,j,d)$ **or**
$DSI^{(t)}(i,j,d{+}1) > \frac{1}{2}DSI^{(t)}(i,j,d)$ **then**
$DSI^{(t+1)}(i,j,d) = 0.8 \cdot DSI^{(t)}(i,j,d)$
else if $DSI^{(t)}(i,j,d{-}1) < \frac{1}{2}DSI^{(t)}(i,j,d)$ **or**
$DSI^{(t)}(i,j,d{+}1) < \frac{1}{2}DSI^{(t)}(i,j,d)$ **then**
$DSI^{(t+1)}(i,j,d) = 0.6 \cdot DSI^{(t)}(i,j,d)$
end if

---

where the neighborhood N consists of the following cells

$$N = \{DSI^{(t)}(i,j,d{-}1), DSI^{(t)}(i,j,d), DSI^{(t)}(i,j,d{+}1)\}. \tag{3.6}$$

This rule introduces a smoothness penalty for neighboring disparity changes, based on the fact that the matching costs between neighboring pixels that belong to the same surface (object) should have small variations. In many cases, however, pixel-wise calculated matching cost values from local window-based stereo correspondence algorithms yield non-unique or wrong correspondences due to textureless areas and ambiguities, leading therefore to wrongly selected disparity values by the similarity accumulator. Assuming that the scene is formed by piecewise-smooth and Lambertian surfaces, the intensity differences $|I_L(i,j) - I_R(i,j)|$, from which the matching costs are computed, should be uniform in smooth areas, while the disparity discontinuities that lie on object boundaries should be aligned with equivalent intensity discontinuities (i.e. strong matching cost variations). Thus, based on the empirically determined factors of the 2nd CA rule, it is possible to smooth or damp the unwanted extreme variations in uniform areas but still keep them discretized in object boundaries and in areas with local gradients.

CA Rule 3 The next CA rule is applied to each element of the DSI with the same disparity level and is defined in Algorithm 1. This rule is applied to a Moore neighborhood of size 5×5 and the variable mod_val represents the number of times that the mode value appears in this neighborhood.

Notice that the main idea behind the previous CA transition rules is that each rule should be applied only to specific places in DSI, where wrong matching costs are evident. Also, the scale factors used by these CA rules were estimated after extensive experiments, based on observations about how the DSI modifications affect the extracted disparity maps and what possible areas may cause false reconstructions. Therefore, these rules target to smoothen the matching costs in a local neighborhood, while it should be also noted that they were explicitly implemented

Algorithm 1 3rd CA rule

$k, p := 0$;
for $m, n := -2$ to 2 **do**
  if $DSI^{(t)}(i+m, j+n, d) \leq \frac{1}{2} DSI^{(t)}(i, j, d)$ **then**
    $k = k + 1$;
  else if $DSI^{(t)}(i+m, j+n, d) \geq \frac{1}{2} DSI^{(t)}(i, j, d)$ **then**
    $p = p + 1$;
  end if
end for
if $k \geq mod\_val$ **then**
  $DSI^{(t)}(i, j, d)$=$0.4 DSI^{(t)}(i, j, d)$
else if $p \geq mod\_val$ **then**
  $DSI^{(t)}(i, j, d)$=$1.2 DSI^{(t)}(i, j, d)$
end if

after extensive testing, in order to produce the maximum possible performance according to the trade-off between accuracy and speed of the proposed technique.

Similarity Accumulator Once the processing of the DSI is completed, we follow a winner-takes-all (WTA) approach and use a similarity accumulator which indicates the most likely disparity value for each pixel (i, j) in the image plane. The selection of the best disparity is performed by searching all disparity levels (from $d_{\min}$ up to $d_{\max}$ in Figure 3.2) for every pixel in the DSI and finding the disparity value for which the refined matching cost $DSI(i, j, d)$ is minimum, i.e.

$$D(i, j) = \arg\min_{d \in [d_{\min}, d_{\max}]} DSI(i, j, d) \tag{3.7}$$

3.5 Hardware Architecture

This section describes the implementation of the proposed stereo reconstruction system in hardware. Generally, the solution of the stereo correspondence problem is a computationally demanding and time-consuming process, mainly due to the repetitiveness of calculations. These calculations cause many memory references, making difficult to meet high performance or real-time speeds, especially for software-based techniques implemented on conventional computers.

Thus, in order to achieve high performance, we designed an efficient parallel-pipelined hardware-based stereo architecture, which implements the proposed

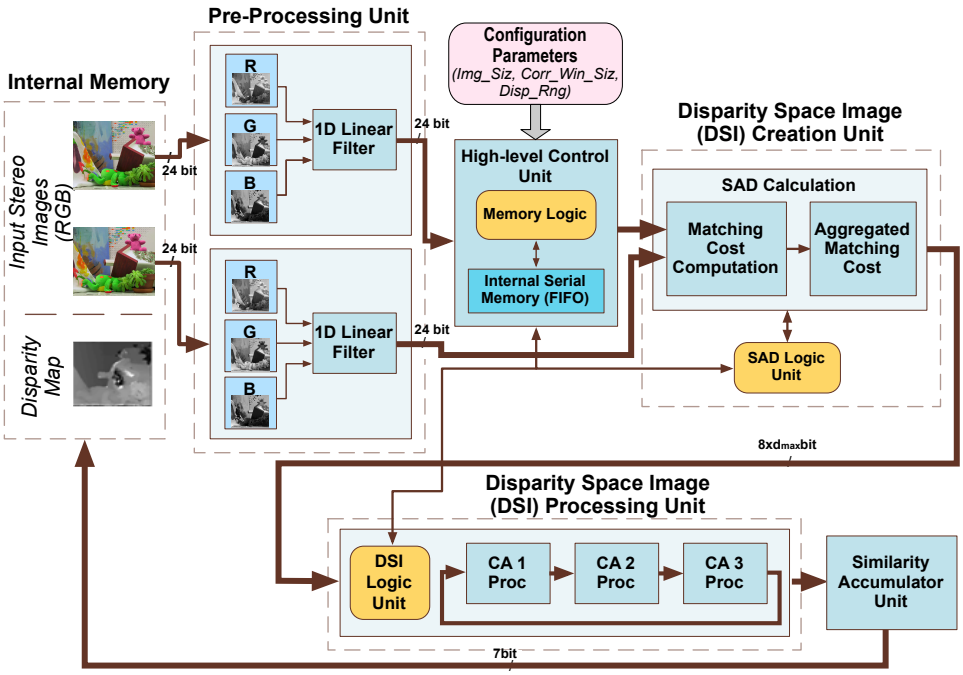


Figure 3.3: *Hardware architecture of the proposed real-time stereo vision system.*

stereo pipeline on a single FPGA device of the Stratix IV family of Altera Devices. The proposed hardware architecture, which is illustrated in Figure 3.3, is fully customized and allows for the parametrization of the input image size, the window size of the correlation matching algorithm (i.e. the SAD) and the levels of the disparity range of the extracted disparity maps.

3.5.1 Overview of Hardware Architecture

The design of the proposed hardware architecture shown in Figure 3.3 follows the stereo reconstruction pipeline described in Section 3.4 and is divided into three main hardware units, the Pre-Processing Unit (PPU), the DSI Creation Unit (DSI_CU) and the DSI Processing Unit (DSI_PU). The system also consists of a High-level Control Unit (HCU), whose task is to coordinate the different operations performed in the architecture, optimize the memory accesses through the Memory Logic module according to the algorithm's requirements, and synchronize the data transfers between the PPU, the DSI_CU and the DSI_PU. This control unit is also responsible for managing the data flow between the internal memory and the different processing stages of the system, in order to reduce the clock cycles needed to load image pixels into the processing units. Furthermore, from this

unit, the user can also select the configuration parameters of the 3D reconstruction procedure, customizing the architectures functionality.

Some of the most important features of the proposed stereo architecture are summarized below.

- **Highly parallel and pipelined implementation.** The proposed stereo system architecture is based on intensive use of parallelism and pipelining design techniques, in order to minimize the area cost of the FPGA implementation and maximize the total throughput of the system. These techniques were implemented without using significant additional FPGA resources, at the expense however of some additional system latency, due to the large amount of programmable registers used.

- **Dedicated processing units.** In our implementation, the main processing parts of the proposed stereo reconstruction method have been realized as simple highly parallel dedicated processing units, in order to achieve real-time throughput. Additionally, all processing units of the system have been designed with a throughput of one pixel per clock cycle, in order to reduce the overall latency.

- **Reduced bus access.** To reduce the unnecessary data flow, the main units of the system were implemented as combined entities, enabling fast data transitions between the elements of each unit and reducing the bus accesses.

- **Multi-buffering operation.** The fast data transition between the different elements of the system is ensured by a multi-buffering data transferring process, which allows the parallel and pipelined processing of data, avoiding writing to memory elements while other components still read from them.

These advantageous hardware design features ensure higher system performance and increased throughput, resulting in generating the final disparity maps in real-time speeds, after a small initial pipeline latency, which is in the order of a few microseconds.

3.5.2 Pre-Processing Unit (PPU) Architecture

As described in Section 3.4.1, we first apply to the input stereo images a weighted mean filter in order to reduce the effects of noise in the stereo matching process. The selection of this filter relies mainly on its efficiency, as it can eliminate adequately the noisy pixels while meeting the requirements imposed by this thesis.

After the acquisition of the image pair from the stereo camera setup, the PPU unit, whose hardware architecture is shown in Figure 3.4(a), stores the input images in the internal memory of the system. The selected FPGA device was carefully selected in order to have enough resources for allowing high-definition images to be stored to its memory components, eliminating the necessity for fetching data from any external memory device.

Next, the input RGB color images are separated through a bus splitter, in order their color components (R, G and B) to be separated and routed to each one-dimensional weighted mean filter, where they will be processed in a parallel manner. The circuitry of this filter can be seen in Figure 3.4(b), in which a combination of shift registers allows the implementation of Equation 3.1. Due to its low resource utilization, we have parallelized this filtering procedure and process the two stereo images simultaneously. This requires the implementation of six identical filters, which are loaded in parallel and after a small delay, their output is also extracted in a parallel manner.

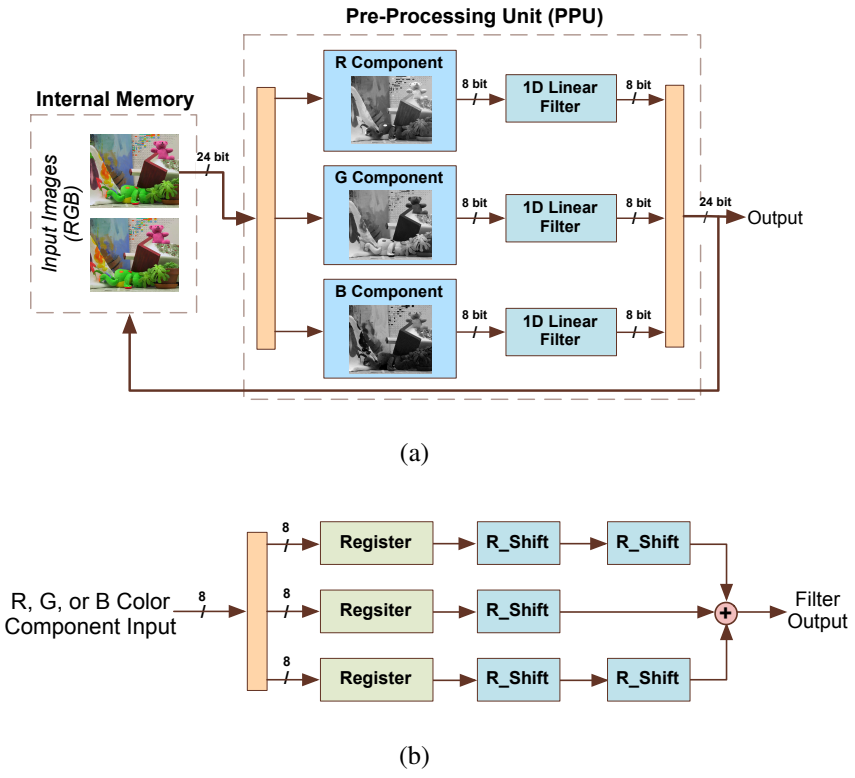


Figure 3.4: *(a) Pre-Processing Unit (PPU), and (b) one-dimensional linear mean filter architecture.*

3.5.3 DSI Creation Unit (DSI_CU) Architecture

After filtering the input images, the accumulated matching costs are calculated by the SAD algorithm and the 3D space of DSI is created, as described in Section 3.4.2. Figure 3.5 shows the proposed hardware architecture of the DSI_CU unit, which is divided into three basic stages, the Color Component Analysis, the Absolute Difference Calculation Module and the Sum of Absolute Differences Module. Each stage corresponds to a dedicated processing module and all of them are managed by the Control Logic unit of DSI_CU. This unit comprises a part of the HCU unit and it is responsible for reading/writing data from/to the different modules of DSI_CU, as well as for coordinating the accesses from/to the internal memory. In the proposed architecture, there is also a Memory Arrangement with two banks of registers (window and scanline registers) for temporarily storing the pixel values needed for the SAD computation process, along with a collection of intermediate pipeline registers/buffers, adders and subtractors to calculate the SAD values.

The general operation of the circuit shown in Figure 3.5 is quite simple and straightforward. First, the two filtered 24-bit color images extracted by the PPU unit are fed into the DSI_CU, where their color components are separated in the Color Component Analysis unit in a similar manner as in PPU. Each color component has 8-bit depth with a total of 256 different intensity values and the Control Logic of the DSI_CU unit reads the pixel data from each component in a row-wise mode, until all data needed to perform the SAD calculation along an entire scanline are fed into the Memory Arrangement.

The DSI_CU Memory Arrangement constitutes a staging area, where all necessary data for the computation of SAD are pre-loaded, in order to avoid unnecessary buffering delays. Considering the default 5×5 window size set in our implementation, three banks of 25+5 8-bit window registers were used in the Memory Arrangement for all three color components of the reference image and a bank of $3{\cdot}(w{+}1){\times}W$ 8-bit scanline registers for the other image.

In the next step, the data loaded in the Memory Arrangement are fed into the SAD Computation Unit, in order for the SAD values to be computed. To reduce the computational cost and the hardware resources utilization of the SAD calculations, the RGB color component data stored in the Memory Arrangement are separately fed to the SAD Computation Unit through a multiplexer arrangement (MUX), whose operation is controlled by the Control Logic of the DSI_CU. This architectural arrangement is very beneficial for the computation of SAD values, since it improves the efficiency of the unit, minimizing the hardware resource utilization and reducing significantly the power consumption. However, following such a design, a small decrease in system's output frame rate is expected, due to the additional delay occurred by loading the pixel data in the multiplexing stage.

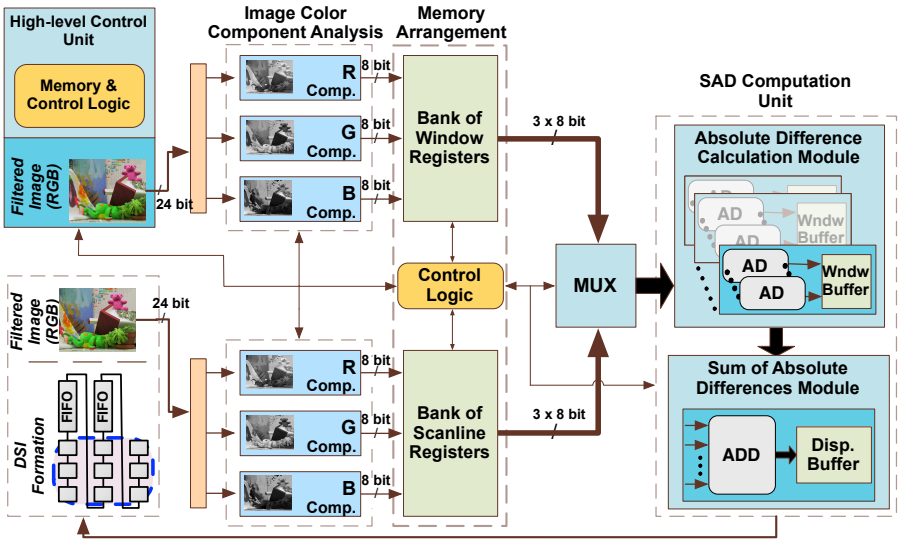


Figure 3.5: *DSI Creation Unit (DSI_CU) architecture.*

Moreover, an additional input signal is fed into the SAD Computation Unit by the Control Logic of DSI_CU, which determines the disparity range for the SAD calculation. In this way, the operating disparity range is customized and the performance of the system may be increased, when small ranges are selected.

Next, the pixel data temporarily stored in Memory Arrangement are fed into the Absolute Differences Calculation module in every clock cycle. This module consists of $d_{\max}$ subtractor modules, which are used for computing in a parallel-pipelined manner the costs of absolute differences between the fixed window of the reference image and the shifting window of the other image. The subtractor modules receive as input $2{\cdot}w^2$ 8-bit data words, with which they compute the absolute differences of pixel intensities and extract 8-bit vectors, which are then added bitwise using the binary tree adders realized in the Sum of Absolute Differences module.

On every clock cycle, the Control Logic unit shifts the current correlation window to the right and feeds the SAD Computation Unit with $3 \cdot w$ 8-bit new values from the bank of scanline registers. The other pixel data from the window and scanline registers remain the same as for the previous operation, and the SAD calculation is performed again for the new pixel data. After $d_{\max}$ right transitions, a new pixel value is added to the bank of window registers and an old one is shifted out in a FIFO (First-In-First-Out) pipeline architecture. Finally, the SAD values are computed for all of the disparity ranges according to Equation 3.4, and the final accumulated cost values are used to form the DSI.

3.5.4 DSI Processing Unit (DSI_PU) Architecture

In this processing stage, the matching cost values computed by the SAD algorithm are refined by a hardware-based parallel-pipelined 2D CA structure as follows.

CA 1 Processing Unit (CA1_PU) Architecture

After the matching cost computation from the DSI_CU unit, the 3D space of DSI has been formed and the first CA rule, whose hardware architecture is shown in Figure 3.6, is applied to each of the $d_{\max}$ disparity image planes of DSI, according to the Equation 3.6. Since it is applied to a 3×3 Moore neighborhood, only 3 scanlines are needed to be loaded initially, while to increase the processing speed and boost the sequential operation of this CA rule, we realized an efficient memory architecture in the early stages of CA1_PU, whose hardware architecture is shown in Figure 3.7.

This memory module uses two FIFO memories to store the $2 \cdot (W-3)$ matching cost values of the first two scanlines (where W denotes the width of the input images) and 9 additional registers to store the 9 elements used by the 3×3 mask of the first CA rule. As the working window of the first CA rule moves over the image, $2 \cdot 3$ overlapping pixels exist between two adjacent windows and 3 pixels change in every iteration. This memory architecture was used to temporarily store these pixels, in order to reduce the clock cycles needed to load the matching cost values into the CA1_PU. Therefore, before the application of the transition rule to the disparity image plane, this memory module needs to be filled up with the first $2W+3$ matching cost values from the DSI_CU unit. Then, in every pixel transition to the right of the disparity image plane, only the bottom right pixel of the working window is new and needs to be fed into the memory module. This allows the output data from the memory architecture to be extracted once per clock cycle, enabling a fast transition of the working window to the next adjacent pixels.

After an initial latency required for filling up the Memory Architecture of the CA1_PU, the first 9 matching cost values that correspond to the matching costs of the first 3×3 window (depicted with 'J1', 'J2', ..., 'J9' in Figure 3.6) are stored in a parallel fashion in equivalent 13-bit registers for being added in the PARAL_ADDER module. Before the PARAL_ADDER, the 9 13-bit signals stored in the registers are converted into vectors of bits from the bitwise analyzers and then the 13 9-bit signals are fed into the PARAL_ADDER, where one addition operation takes place in every clock cycle. Although we could use for the addition operation the embedded Adder modules provided by the FPGA device, we preferred to implement a bitwise fully parallelized adder (PARAL_ADDER), since typically the embedded modules in FPGAs consume more power than the bitwise logical operations, mainly due to their architectural design and longer instruction

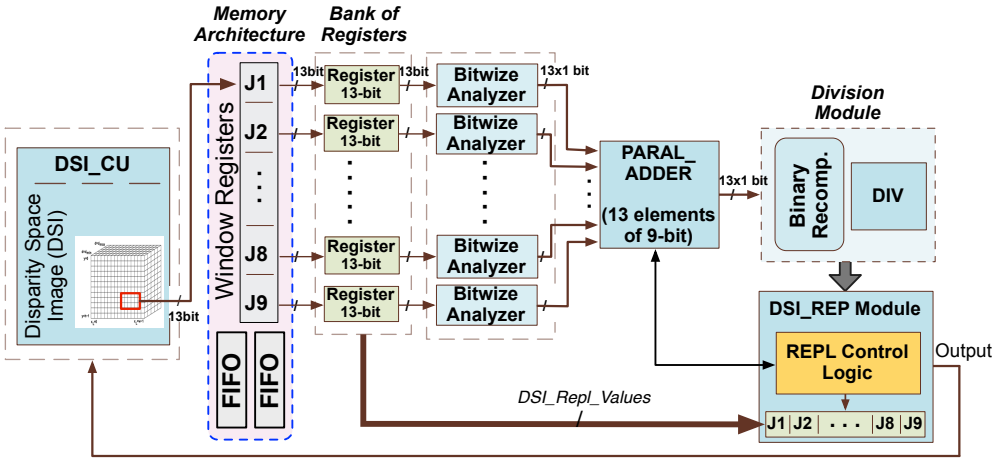


Figure 3.6: *Hardware architecture for the first CA processing unit (CA1_PU).*

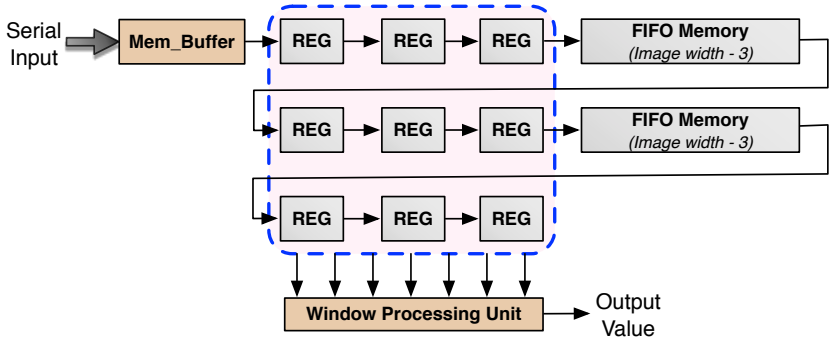


Figure 3.7: *Memory architecture for CA1_PU.*

pipelines.

Eventually, after performing the addition operation, the resulting summation value is recomposed to a 13-bit signal which is next divided by the Division module. Its quotient is then fed into the DSI_REP module, where the REPL Control Logic synchronizes the replacement of the matching cost value of the central pixel of Moore's neighborhood with the refined value extracted by CA1_PU. Finally, all these values are fed into the CA2_PU unit for further refinement.

CA 2 Processing Unit (CA2_PU) Architecture

As opposed to the first CA rule, the second CA structure refines the matching cost values along the d axes of DSI, where three directly adjacent neighboring pixels

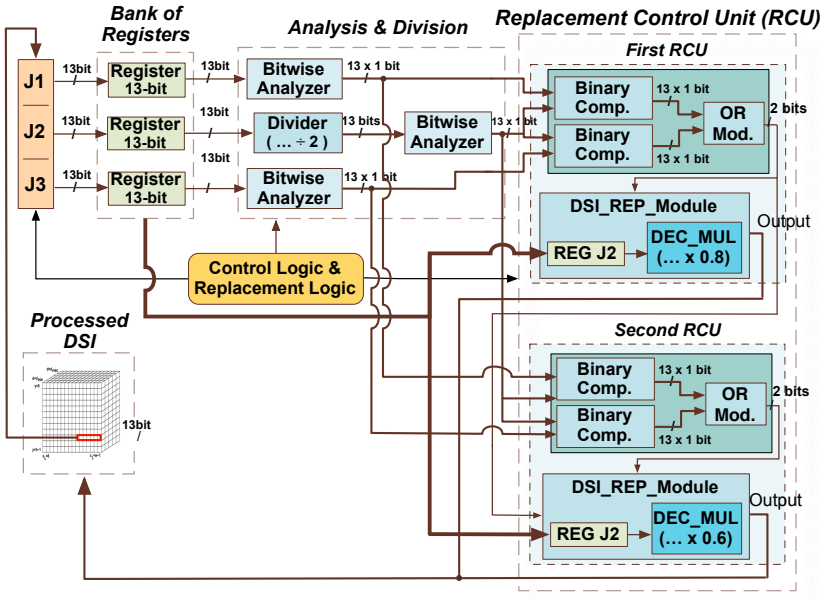


Figure 3.8: *Hardware architecture for the second CA Processing Unit (CA2_PU).*

$(d-1, d, d+1)$ are needed for each comparison.

Figure 3.8 shows the parallel-pipelined architecture we realized for CA2_PU, which consists of four main tasks, i.e. the DSI value fetching, matching cost analysis & division, matching cost comparison and DSI replacement. There is also a central Control & Replacement Logic module, which is part of the main HCU of the system and its role is to coordinate the operations performed by all modules in CA2_PU, while there is also an on-chip memory block (bank of registers) for temporarily storing the matching cost values of DSI.

To feed into the CA2_PU unit the three necessary pixel values for initiating the execution of the second CA rule (depicted as 'J1', 'J2' and 'J3' in Figure 3.8), three adjacent disparity image planes from DSI should be buffered. To achieve this in an efficient manner, we utilized a custom DSI Memory Block (see Figure 3.9), which consists of only two image plane buffers along with four registers. After $2 \times W \times H + 2$ clock cycles, an input register (In_Register) and two image plane buffers are loaded sequentially with the DSI values from the first two disparity image planes. Since the CA2_PU unit requires three input pixels with same coordinates in the $x-y$ plane and different disparities from three neighboring disparity image planes (e.g. pixels $P(i, j, d)$, $P(i, j, d-1)$ and $P(i, j, d-2)$), the values of the three left top-most registers of the DSI Memory Block could be used to feed this unit. These three registers, whose values correspond to the input values of

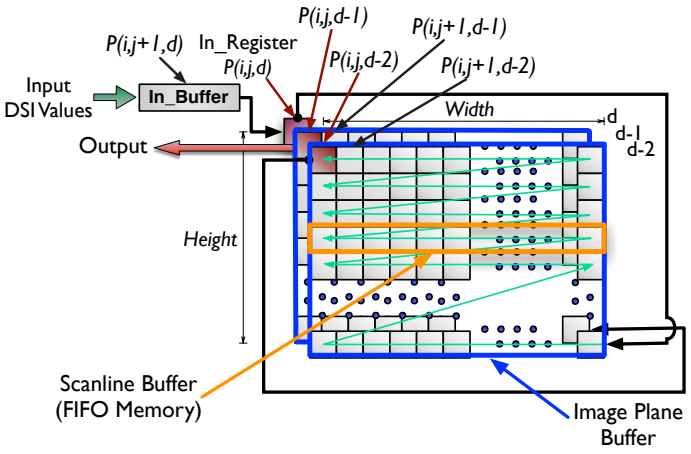


Figure 3.9: *Hardware architecture for the DSI Memory Block.*

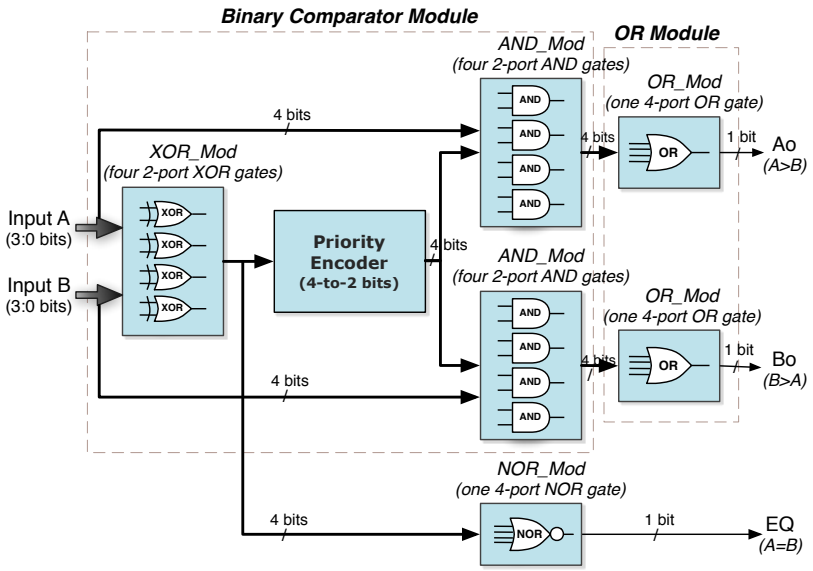


Figure 3.10: *The implemented binary comparator.*

CA2_PU, are depicted with cherry red color in Figure 3.9,

After feeding the CA2_PU unit with the required pixel values from DSI, the input values are stored temporarily into an on-chip memory block (bank of registers) and are then inserted synchronously into the Analysis & Division module, where similarly to the architecture design of CA1_PU, their signals are converted

to vectors of bits and are forwarded to the first Replacement Control Unit (RCU).

In our architecture, there are two Replacement Control Units (RCUs), which implement the first and the second part of the second CA rule, respectively. Each of them uses two Binary Comparator modules for comparing the middle with the other two 13-bit input signals, while there is also one logic OR module in RCUs for the final replacement decision.

To perform the binary comparisons, which is one of the most computationally demanding tasks in the proposed stereo system, we designed a hardware-efficient comparator for comparing 2 13-bit input signals. Figure 3.10 shows a simplified architecture of the proposed Binary Comparator implementation, using a comparison example with two 4-bit input signals, in order to demonstrate better the functionality of the proposed architecture. As it can be seen, its circuit architecture is quite straightforward, since it uses only logic gates for its implementation, and no complex calculations such as multiplications. This beneficial architecture for such a complex computational unit substantially reduces the hardware complexity, making easy to scale up for implementing larger comparators with more inputs, while it improves also the total efficiency of the system, contributing to faster 3D reconstructions. Moreover, the reduced size of this circuit allows for a parallel implementation of more than one comparators, giving the ability of comparing more than two pixel values at a time.

In the final stage of the CA2_PU unit, the output signals of Binary Comparators are compared in the OR modules and then are driven into the DSI_REP modules, in order to be multiplied and replaced by the proper refined values.

CA 3 Processing Unit (CA3_PU) Architecture

The third CA rule of the proposed DSI refinement process initiates its operation by comparing the 25 pixel values of a 5×5 Moore neighborhood in the same disparity image plane with the neighborhood's central pixel, and the comparison results are summed for further comparison with the number of times (frequency) the mode value is presented in the Moore's neighborhood.

In Figure 3.11, the fully parallel-pipelined architecture of the CA3_PU unit is presented, which is separated into six different stages: DSI value fetching, matching cost Analysis & Division, mask matching cost value comparison and counting, and DSI replacement. In addition, similarly to CA2_PU, the architecture includes two Replacement Control Units (RCUs) and a Control & Replacement Logic module with similar functionality as that of CA2_PU. Moreover, an on-chip memory block consisting of a bank of registers is also included in the early stages of CA3_PU for temporarily storing the input matching cost values.

In the initial operating stage, a memory module with similar architecture as the one shown in Figure 3.7 was realized, including 4 FIFO memories and 25 registers

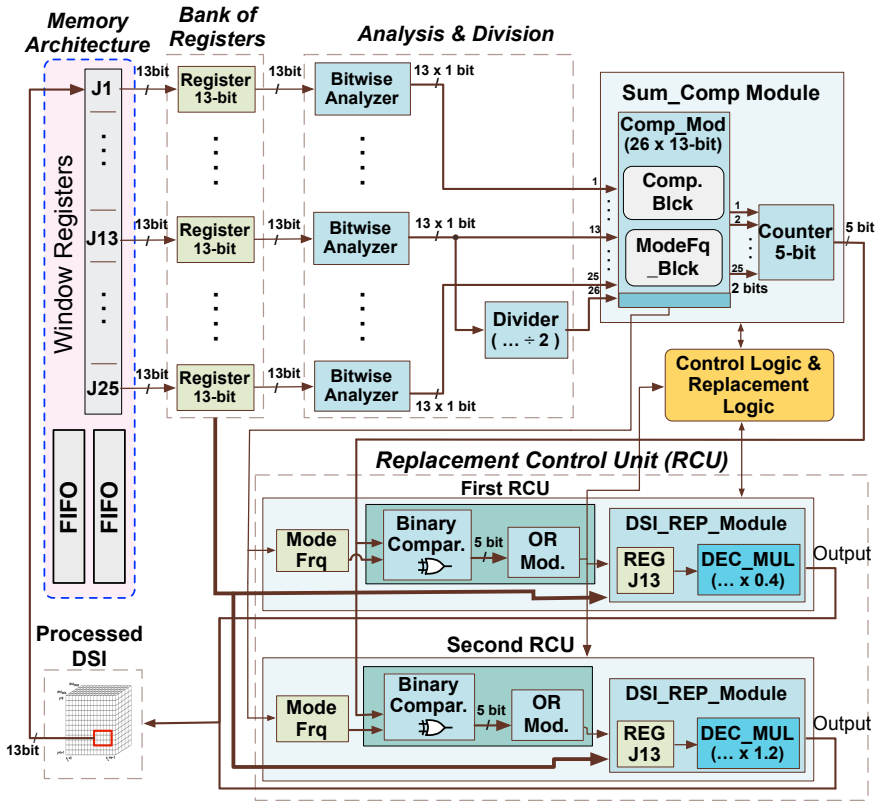


Figure 3.11: *Hardware architecture for the third CA Processing Unit (CA3_PU).*

for buffering all necessary pixel values from DSI. After an initial latency for filling up this memory module, the 25 image pixels of the 5×5 working window are stored simultaneously in an input register arrangement and after being converted into multi-line signals in the Analysis & Division stage, they are forwarded to the Sum_Comp module.

In this module, the Comp_Mod module performs the comparison between the neighboring pixel values of the mask with the central pixel of the working window (Comparator_Blck) using the hardware architecture presented in Figure 3.10 with 25 13-bit input signals and equivalent 2-bit outputs. In addition, the frequency of the mode value is also computed in the Comp_Mod module, using a separate circuit (ModeFq_Blck module) whose architecture is shown in Figure 3.12. In this processing block, each Neighb_Comp sub-block constitutes a separate logic comparator, which compares one input pixel value with the other 24 of the 5×5 neighborhood, and if this value stands more than 12 times inside the neighborhood, then the output of Neighb_Comp is assigned as the logic one. The priority encoder generates a signal indicating which Neighb_Comp block contain the mode

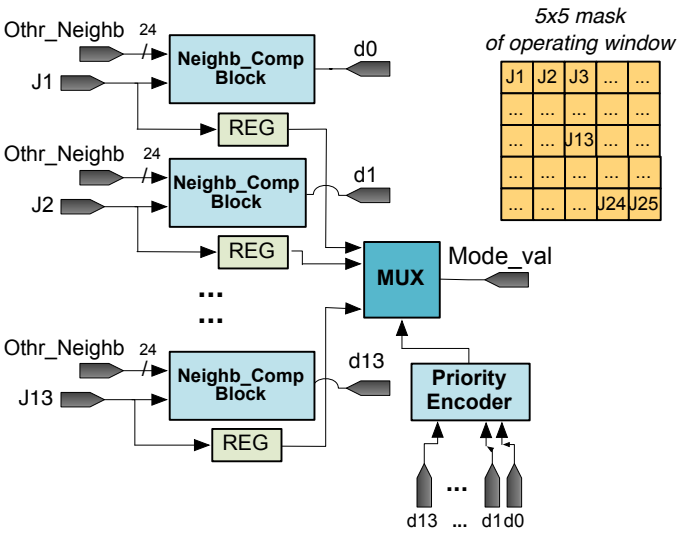


Figure 3.12: *Hardware architecture for the ModeFq_Blck sub-module.*

value, and this index value is fed into the multiplexer (MUX), which selects the mode value of the neighborhood.

After an initial latency of 4 clock cycles, the output of this sub-module appears once per clock cycle and is fed into the next pipeline unit (i.e. the RCU unit), in order to be compared with the output of the counter. In the final stage, the RCU implements the outer comparisons of the condition parts of third CA rule and the resulting product of the DSI_REP module is fed into the next processing stage, as depicted in Figure 3.3.

Similarity Accumulator Architecture

Once the DSI refinement process has finished, the refined pixel values are fed into the Similarity Accumulator Unit, which selects the proper disparity value for each pixel in the image plane, as indicated by the Equation 3.7. The hardware architecture we used for implementing this unit relies on an insertion sort network, similar to the one presented in [Mueller et al., 2012].

3.6 Results

In this section, we demonstrate the effectiveness of the proposed stereo reconstruction system by evaluating it on the Middlebury stereo evaluation benchmark, which is a widely used test procedure for the taxonomy and comparison of two-frame stereo correspondence algorithms. All datasets used for the evaluation of

our method include scenes from indoor environments, which present different levels of complexity and highlight the capabilities of the proposed method. We also compare our novel hardware-based stereo reconstruction architecture against other similar state-of-the-art approaches, indicating the benefits of our system.

Implementation The hardware architecture presented in this work has been implemented on a custom FPGA device of the Altera Stratix IV family. For the development of the schematic design we used the Altera Quartus II graphic editor, which allows to implement circuit designs in FPGA devices. The resource utilization report exported by Altera Quartus II for the proposed FPGA implementation is summarized in Table 3.1.

In this table, the overhead caused by the DSI_CU and DSI_PU units is presented, since these are the most demanding processing units of our system and present the highest realization complexity against the other units. We note that these results were extracted keeping the image size, the window size and the disparity range constant, while for this general overview, the disparity range was selected to have a typical mid-range value of 70 disparity levels. It is also worth noticing the low device resource utilization of the proposed architecture, which was mainly achieved by its reduced design complexity and the multiplexing schema in the DSI_CU unit. Furthermore, under the configuration presented in Table 3.1, the operating clock frequency of our system was found to be 168 MHz, while for different configurations, the clock frequency varies as indicated in Table 3.8.

It should be also mentioned that in our system, the size of the input stereo image pair does not affect the device utilization directly or in run-time, since in our approach we use only local (i.e. window-based) image processing methods, which do not modify the image size. However, the size of the input image is directly related to the size of the on-board memory and affects the total latency/delay caused by the data transfers between the different units of the system. In addition, the operating disparity range and the window size of SAD linearly affect the consumed logic resources, since both of them are related to the memory utilization of the system (see also Tables 3.6, 3.7 and 3.8 for more information). Moreover, from the results presented in Table 3.1 can be also inferred that the proposed hardware implementation still leaves enough resources in the targeted FPGA device for additional implementations, which could further improve the system's performance and extend its capabilities.

Quantitative and qualitative evaluation The test benchmark we used for evaluating our novel stereo reconstruction method relies on the widely used evaluation procedure reported by Scharstein and Szeliski [Scharstein and Szeliski, 2002], which is available at vision.middlebury.edu/stereo. This benchmark consti-

Table 3.1: *Resource utilization of target device.*

Device (Altera Stratix IV)	**Total Registers (%)** (424,960)	**Logic Utilization (%)** (424,960)	**Total ALUTs (%)** (424,960)	**Total LABs (%)** (21,248)	**Total Pins (%)** (1,112)
DSI_CU	15,2	14,7	4,9	6,3	4,4
DSI_PU	9,8	6,8	6,3	4,1	3,5
Total	25,6	23,2	12,6	11,4	8,5

tutes a reference point for the majority of stereo reconstruction methods, since it includes a large number of multi-exposure and multi-lighting stereo datasets from various environments that present a varied set of challenges. Notice, however, that due to the wide range of methods reported to the Middlebury test procedure, many of them, although being state-of-the-art in accuracy, they have not been included in our comparison, since our work focuses mainly on real-time or near real-time hardware-based methods, while these methods are far from such requirements.

The evaluation data that were used as benchmarks from Middleburry's database are calibrated stereo image pairs and consist of standard as well as some new Middlebury stereo image pairs. In our setup, we applied fixed-value boundary conditions for all window-based algorithms used in our reconstruction pipeline, in which the transition rules were only applied to non-boundary cells.

Figure 3.13 shows the resulting disparity maps produced by our system for different stereo image pairs of varying image sizes and different disparity ranges, all obtained by the Middleburry database. As can be seen, the generated disparity maps provide adequate disparity accuracy, considering the real-time performance and despite the challenging indoor environments which present increased level of detail and many ambiguous areas. Note also that the strip on the left side of the extracted range images was caused by the disparity between the input stereo images, while its width is directly proportional to the operating disparity range of each dataset.

Figure 3.14 provides a qualitative evaluation of our extracted results against other related architectures, although many authors do not provide quantitative and/or qualitative results for their disparity maps. Nonetheless, the results in Figure 3.14 confirm the performance of our reconstruction method over the related state-of-the-art techniques. For example, in the disparity map from Tsukuba image pair, the table and the area around it are more accurately reconstructed with our method than with other approaches, while the depth values at object discontinuities are also better reconstructed. The camera on the tripod constitutes another example, where the results of our method outperform the other results. Despite,

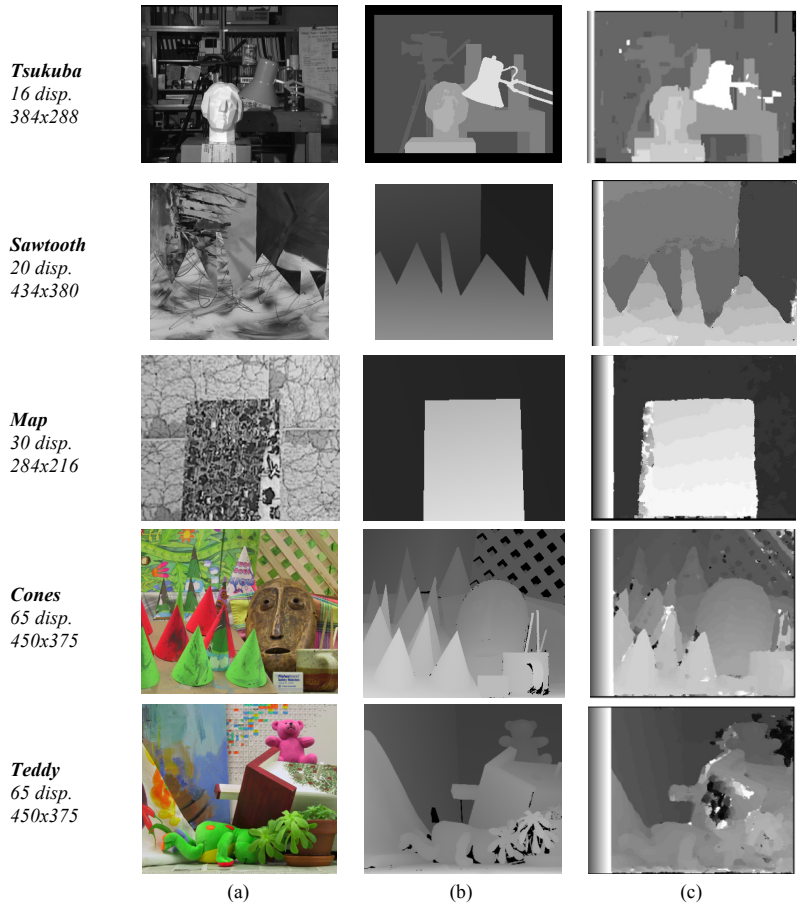


Figure 3.13: *Resulting disparity maps of stereo image pairs: Tsukuba, Sawtooth, Map, Cones and Teddy. (a) Input images, (b) ground truth and (c) final disparity maps.*

however, presenting higher accuracy from the other techniques, some errors still remain in the disparity maps produced by our system, which are caused either by the wrong replacements of CA rules, or by mismatches due to the SAD algorithm.

Table 3.2 shows the quantitative results regarding the accuracy of our reconstruction pipeline under various configurations. The evaluation metric we used to measure the reconstruction error in the extracted disparity maps was the error rate ε, which indicates the percentage of bad pixels whose absolute disparity error is greater than 1. It was calculated with respect to the ground truth maps from the

Table 3.2: *Quantitative results of the proposed system, regarding accuracy under various configurations.*

	Disparity range	**Error ε (%)**
Tsukuba (384×288 pxl)	16	9.1
Sawtooth (434×380 pxl)	20	10.7
Map (284×216 pxl)	30	17.3
Cones (450×375 pxl)	65	24.7
Teddy (450×375 pxl)	65	25.4

Middleburry evaluation database using the cost function:

$$\varepsilon = \frac{1}{N} \sum_{i,j} (|d_{\text{comp}}(i,j) - d_{\text{ground}}(i,j)| > 1) \tag{3.8}$$

Hardware performance evaluation Besides the qualitative and quantitative evaluation of the disparity maps extracted by the proposed stereo reconstruction system, its hardware performance and efficiency has been also evaluated. Tables 3.3 and 3.4 present the performance evaluation results of our system architecture compared to other stereo vision systems with similar window matching techniques or with other disparity computation methods, respectively. Due to the configuration variabilities presented in these systems, we provide also the related configuration parameters, such as the image size and the disparity range. Additionally, for evaluation purposes we compute a normalized performance index (NPI) in terms of $image\_size \times disparity\_range \times fps/frequency$, which provides a more general overview for the performance of the systems under consideration.

As shown in Tables 3.3 and 3.4, our hardware implementation presents improvements over the majority of previous systems in terms of performance and processing rates. In addition, although there are some stereo systems that achieve better performance indices (NPI) than our method, such as the systems from [Lee et al., 2005; Ambrosch et al., 2009] (in Table 3.3) and [Sabihuddin et al., 2008] (in Table 3.4) other factors, such as the different configuration parameters, the total amount of calculations per second (i.e. processing rate) or the complexity-accuracy and quality-performance trade-offs should also be taken into account. For example, from Table 3.3, our hardware architecture produces 640×480 disparity maps with 70 disparity levels and 114 fps, which makes a total amount of about 2.45 billion disparity calculations per second. Considering however the data presented in Table 3.4 for the same parameters, arises that our architecture

Table 3.3: *Performance evaluation results of the proposed system compared to other related hardware-based stereo vision systems with similar window matching techniques.*

	Implemented Device	Matching Method	Image Size	Disparity Range	Window Size	Frequency (MHz)	Frames per Second (fps)	Normalized Performance Index
[Hile and Zheng, 2004]	N/A	SAD	521×480	32	N/A	N/A	30	N/A
[Miyajima and Maruyama, 2003]	FPGA (Xilinx)	SAD	640×480	80	5×5	40	26	15.97
[Arias-Estrada and Xicotencatl, 2001]	FPGA (Xilinx)	SAD	320×240	16	7×7	66	71	1.32
[Lee et al., 2005]	FPGA (Xilinx)	SAD	640×480	64	32×32	10	30	58.98
[Hariyama et al., 2005]	FPGA (Altera)	SAD	64×64	64	8×8	86	5063	15.43
[Kuhn et al., 2003]	ASIC	SSD / Census	256×192	25	Census & Corr.	75	50	0.81
[Ambrosch et al., 2009]	FPGA (Altera)	SAD	450×375	100	9×9	110	600	92.04
[Woodfill et al., 2004]	ASIC	Census	512×480	52	N/A	60	200	42.59
[Jia et al., 2004]	FPGA (Xilinx)	SSAD	640×480	64	N/A	N/A	30	N/A
[Gudis et al., 2012]	FPGA (Altera)	Pyramid decomposition	$2k$×$2k$	N/A	N/A	200	N/A	N/A
Proposed Arch.	FPGA (Altera)	SAD	640×480	70	5×5	168	114	14.59

can make around 1.85 billion calculations per second. This difference indicates that relying only on these parameters for comparing the performance of a system is not reliable and other factors should be taken also into consideration. Furthermore, other architectures that their performance index (i.e. NPI) outperforms ours several times, such as the systems in [Lee et al., 2005] (in Table 3.3) and [Sabihuddin et al., 2008] (in Table 3.4) which present almost 4 and 3 times higher NPIs than ours, can make a total amount of almost 590 million and 2.5 billion calculations per second, respectively, which shows that our system can process almost 4 times more data than the system in [Lee et al., 2005] and only about 25% less than the system in [Sabihuddin et al., 2008].

In addition, our stereo reconstruction architecture maximizes the trade-off between accuracy and performance, producing more accurate disparity maps than almost all other techniques with higher NPI or processing rates. For example, although the system in [Ambrosch et al., 2009] (in Table 3.3) outperforms our method several times, the error rate of its extracted disparity maps for the Teddy

Table 3.4: *Performance evaluation results of the proposed system compared to other hardware-based stereo vision systems.*

	Device	Image Size	Disp. Range	Frames/sec	Norm. Perform. Index
Archit. in [Masrani and MacLean, 2006]	FPGA (Altera Stratix S80)	640×480	128	30	N/A
Archit. in [Díaz et al., 2007]	FPGA (Xilinx Virtex II)	1280×690	9, 15, 29	52	14.95
Archit. in [Lu et al., 2007]	GPU (Nvidia GeForce 7900)	512×512	64	38	N/A
Archit. in [Sabihuddin et al., 2008]	FPGA (Xilinx Virtex II Pro)	640×480	128	63.54	30.96
Archit. in [Banz et al., 2010]	FPGA (Xilinx Virtex V)	640×480	128	30	8.86
Archit. in [Hadjitheophanous et al., 2010]	FPGA (Xilinx Virtex II Pro)	640×480	25	8.9	0.68
Archit. in [Pérez et al., 2009]	FPGA (Xilinx Virtex V)	1280×720	96	2.5	1.1
Archit. in [Jin et al., 2010]	FPGA (Xilinx Virtex IV)	640×480	64	230	18.41
Proposed Archit.	FPGA (Altera Stratix IV)	640×480	128	47	11.27

image pair is almost 44% while ours 25%, i.e. 43.1% less. Moreover, if we compare our results from the Tsukuba image pair with the results extracted by the systems in [Lee et al., 2005] and [Sabihuddin et al., 2008], we can see that their results are more cluttered and the object boundaries are more distorted than the results extracted from our stereo reconstruction method. Note also that their results suffer from inter-scanline inconsistencies, which is a typical error presented in local-based stereo reconstruction algorithms, while our technique eliminates these errors by its efficient DSI optimization process. This proves that our stereo reconstruction method exhibits high potentials for real-time applications, where tight time constraints and increased accuracy are necessary.

Performance impact of system parameters We have also evaluated the performance of our system under various configurations, indicating how its overall speed and output frame rate are affected by the input parameters. Table 3.5 shows the impact of the input image size on the performance of the proposed system, keeping the disparity range and window size constant. As it can be observed, the performance of the system (i.e. output frame rate) decreases as the image size increases, although the overall frame rate of our architecture approximates real-time speeds (i.e. 30 fps) even for high-definition images with size 1280×1024.

The image size is also related to the I/O bandwidth of the architecture, which

affects the performance of the proposed system, mainly when an external data resource (e.g. memory) is used. In this case, the delay to fill the scanline buffers with the temporarily stored image pixels needed for the window correlation is almost proportionally connected to the bandwidth of the system. Similarly, the performance of the system is also decreased as the I/O bandwidth decreases, since the data flowing into the system limits its throughput.

Considering the impact of disparity range on system's performance, Table 3.6 shows the resource utilization results and the operating frequency of our architecture for different disparity levels up to 128. As it is shown, there is an almost linear increase in the utilization of the FPGA resources, which is caused mainly from the additional circuit elements used for calculating the SAD values. Furthermore, the variations in operating frequency caused by the changes in the disparity levels indicate that when the disparity range increases, the frequency decreases, meaning that the hardware overhead becomes also higher at high disparity ranges.

Similar results are presented in Table 3.7, where the impact of specific window sizes on the system's resource utilization is shown. As it can be seen, keeping the disparity range and image size constant while increasing the window size leads also to an increase in the utilization of the FPGA. This is justified by the fact that more image lines are needed in every processing stage to be fed and stored to the bank of scanline registers and to the memory arrangements, in order the SAD values and the results from each CA processing unit to be computed. Table 3.8 illustrates the system's resource and performance results for different image size configurations while window size and disparity levels remain the same. We notice here that the results show a similar behavior as in the previous cases, i.e. an increase in image size leads to a decrease in system's frequency and to a parallel increase in system's resource utilization, since the image size affects strongly the amount of hardware components used in the system. However, the system clock frequency is slightly affected in this case, having only a small decrease as the size of input images increases, meaning that our system is capable to keep its high performance even when high-definition range images are generated.

3.7 Conclusion and Discussion

In this chapter, we presented a new hardware-based stereo reconstruction method, capable to produce dense disparity maps in real-time, targeting mainly time-critical 3D reconstruction applications. The reconstruction pipeline consists of a fast local-based stereo matching algorithm for the formation of DSI and an efficient DSI optimization technique, which relies on a highly effective CA structure. Despite their simple construction, CA have shown that they are capable of highly complex behavior, due to their inherent parallelism and local interconnection, and

Table 3.5: *Frame rate output (fps) of the proposed architecture.*

	System's unit performance		
Image Size (pixels)	**DSI_CU**	**DSI_PU**	**Overall FPGA System**
320×240	1467	461	455
640×480	398	120	114
800×600	259	79	73
1024×768	163	50	45
1280×1024	98	32	27

Table 3.6: *Resource utilization for different disparity level configurations.* (Image size = 1024×768, window size = 7×7)

Disparity range	**20**	**40**	**70**	**128**
Total registers(%)	22,7	23,4	24,1	25,2
Logic Utilization(%)	18,6	19,4	20,5	22,3
Frequency (MHz)	177,2	174,3	168,1	156,4

Table 3.7: *Resource utilization for different window size configurations.* (Image size = 1024×768, disparity range = 70)

Window size	**3×3**	**5×5**	**7×7**	**9×9**
Total registers(%)	22,8	23,5	24,1	24,8
Logic Utilization(%)	19,5	19,9	20,5	21,4
Frequency (MHz)	186,1	178,7	168,1	154,7

therefore they were able to efficiently refine the wrong matching cost values of DSI, improving substantially the extracted disparity images.

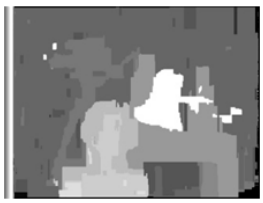
We also implemented the proposed reconstruction pipeline on a single FPGA device using a fully parallel-pipelined architecture, in order to meet the timing and processing constraints for real-time applications. The proposed hardware architecture presents high performance and increased scalability, ensuring increased

Table 3.8: *Resource utilization for different image size configurations.*
(window size = 7×7, disparity range = 70)

Image size	320×240	640×480	800×600	1024×768	1280×1024
Total registers(%)	21,2	22,3	23,2	24,1	25,4
Logic Utilization(%)	17,6	18,5	19,2	20,5	22,7
Frequency (MHz)	172,1	171,4	170,1	168,1	165,7

output frame rates and showing an output latency depending mainly on the configuration parameters of the system.

Despite its novelty and efficiency, our stereo reconstruction approach presents also some important limitations. Our method uses a local-based stereo matching algorithm for computing the matching costs of corresponding pixels, mainly due to its simplicity and practical usage. However, this method (and in general the local-based stereo methods) introduces many mismatches and is prone to errors, since it relies on pixel-based intensity comparisons, which are highly affected by various factors that can distort the pixel intensities and deteriorate the quality of the computed disparity maps. Therefore, a global-based stereo matching algorithm could produce better results, at the expense of course of increased computational complexity. However, due to the recent advances in custom hardware technology, it might be possible such an algorithm to be efficiently implemented in a custom hardware device, allowing the generation of more accurate disparity maps with fast processing speeds. Moreover, although the CA-based refinement stage of our pipeline is quite efficient and can identify many mismatches, it could be potentially improved by optimizing further the CA rules, since CA present rich computational capabilities and can be used efficiently for modeling more complex behaviors.



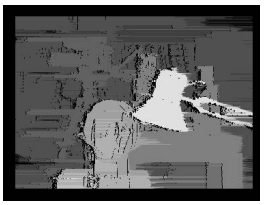
(a) Proposed Arch. (Acc 90.9%)

(b) Arch. in [Miyajima and Maruyama, 2003] (Acc N/A)

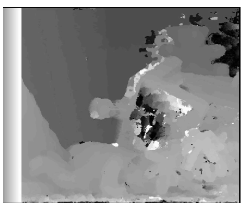
(c) Arch. in [Hariyama et al., 2005] (Acc N/A)

(d) Archi. in [Lu et al., 2007] (Acc 88.44%)

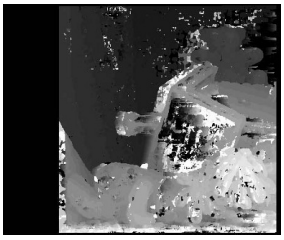
(e) Archi. in [Sabihuddin et al., 2008] (Acc N/A)



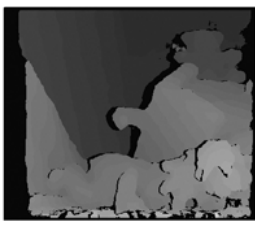
(f) Archi. in [Banz et al., 2010] (Acc N/A)



(g) Proposed Arch.(Acc 74.6%)



(h) Archi. in [Hile and Zheng, 2004] (Acc N/A, 5x5 mask)

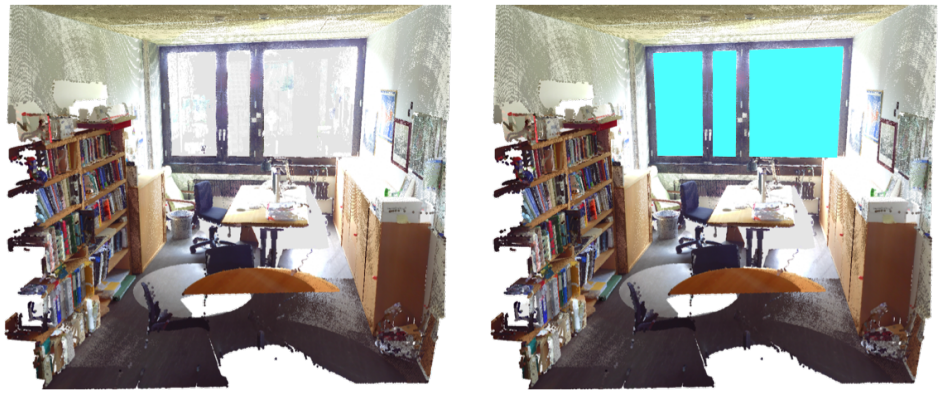


(i) Archi. in [Lu et al., 2007] (Acc 78.50%)

Figure 3.14: *Qualitative evaluation for Tsukuba and Teddy stereo image pairs for different approaches.*

C H A P T E R 4

WALL OPENINGS RECONSTRUCTION



4.1 Introduction

In chapter 3, we described the first research contribution of this thesis, which focuses on the acquisition stage of the reconstruction pipeline. Our second contribution, which is presented in this chapter and contributes to the corresponding annotated parts of Figure 1.2, focuses on the 3D modeling stage of the reconstruction process and presents a method for the accurate detection, reconstruction and semantic segmentation of the architectural wall elements of building interiors.

4.1.1 Motivation

The identification of the architectural wall openings in indoor environments is very important for the recovery of the building's *as-is* state. Although most of the times a raw blueprint with wall indications and other rough floorplan information is provided, the actual information about the building's architectural wall elements, such as the windows or door locations, is not always available or may not be up-to-dated from the initial design time.

Thus, indoor modeling methods should be capable to reconstruct this valuable and useful information automatically, in order to generate valid and reliable BIM models that could be used in a variety of applications. Thus, the problem of reconstructing the architectural wall elements of building interiors is explored in this work, since it is still very challenging and is only partially solved by the recently proposed methods. This is due to the high complexity of the problem, since the majority of real-world indoor environments are typically cluttered with objects and obstacles, which prevent the perfect acquisition of the scene (see also Figure 4.1). In addition, erroneous model interpretations due to complex room layouts and highly occluded areas affect also the accuracy of scene reconstruction, especially when occluders hide significant parts of the wall surfaces and their wall openings. Furthermore, windows and other highly reflective surfaces can drastically affect the acquired models by introducing large-scale artifacts and causing missing data, due to their unfavorable material properties. Windows in particular, which are mainly shapeless, textureless and transparent, constitute an important source of reconstruction errors and artifacts. Moreover, doors are usually hard to be detected, since their reconstruction depends strongly on the wall geometry and the descriptive features of the door.

Despite the recent research efforts, a satisfactory automatic approach for reconstructing the architectural wall elements of building interiors has not been presented yet. State-of-the-art methods rely mainly on strict assumptions about the scene and present many restrictions and limitations to their applicability. Some of them provide evidences for the wall elements relying on a *a priory* knowledge obtained by complex and time consuming machine learning algorithms [Adan and

Figure 4.1: *Examples from heavy occlusions and clutter that can occur in indoor environments (occluders are still visible and some of them are also marked).*

Huber, 2011; Zhang and Zakhor, 2014; Xuehan et al., 2013], or they combine raw data from indoors and outdoors in order to eliminate occlusions [Previtali et al., 2014]. Other pipelines consider occlusion-free raw data or eliminate occlusions by scanning indoor scenes with only the essential interior objects and without furnitures [Budroni and Böhm, 2010b]. More recent reconstruction pipelines combine color and/or stereo images along with 3D range data, in order to identify and segment the building wall elements [Valero et al., 2012; Dumitru et al., 2013; Díaz-Vilariño et al., 2014; Stambler and Huber, 2014; Turner et al., 2015; Sui et al., 2016]. Thus, a method capable to reconstruct the architectural wall elements of building interiors automatically, avoiding the restrictions and limitations introduced by the current approaches, would be very beneficial for the 3D modeling of indoor environments, especially if it could be embedded as an additional processing and refinement module to the current modeling pipelines.

4.1.2 State of Research

Although the automatic generation of indoor models is a very active research topic, only few approaches try to address the problem of the wall elements reconstruction [Adan and Huber, 2011; Ochmann et al., 2016]. A comprehensive review about the building reconstruction methods can be found in [Tang et al., 2010; Volk et al., 2014]. Since this contribution focuses more on the reconstruction and modeling of the architectural wall elements, we assume that the architectural building

model has been already reconstructed (see also Figure 1.2). Thus, in this section we summarize only the most relevant approaches that can either reconstruct the architectural wall elements, or they can generate a faithful architectural building model that could be used as input to our pipeline.

The last few years, many works have been proposed for the automatic reconstruction of building interiors [Okorn et al., 2010; Kotthauser et al., 2013; Mura et al., 2014; Oesau et al., 2014; Adan et al., 2015]. Some of them [Okorn et al., 2010] create a representation of the floor plan and identify the wall line segments in Hough space. Other methods [Kotthauser et al., 2013; Mura et al., 2014] allow a flexible and occlusion-aware reconstruction of wall surfaces by finding the wall planar patches and creating a polyhedral representation of the building structure, while others [Adan et al., 2015] use mobile robots to optimize the acquisition of 3D space and then detect the rooms' contours, from which the wall surfaces can be extracted.

The automatic reconstruction of architectural wall elements has also received significant development the last years, and the methods proposed so far vary greatly depending on the information they exploit. In most recent approaches, such as the one presented in [Ochmann et al., 2016], a global optimization algorithm is used to reconstruct the building interior and the wall features are then identified by a greedy clustering algorithm. Despite its efficiency, this method presents limited applicability, since it requires low-cluttered environments and certain conditions under which it can be applied, while its wall element reconstruction approach can also fail in some fairly common cases, e.g. in the case of multiple neighboring windows. A more restrictive approach is described in [Previtali et al., 2014], where the wall elements can be reconstructed only under the combination of raw data from indoor and outdoor environments. In [Stambler and Huber, 2014], an enclosure reasoning is used to obtain the best coverage of the most likely rooms, while the wall openings are explicitly detect by a region growing algorithm.

Color or depth images along with 3D point clouds have been also combined by many approaches [Adan and Huber, 2011; Budroni and Böhm, 2010b; Xuehan et al., 2013; Dumitru et al., 2013; Díaz-Vilariño et al., 2014], in order to faithfully model the environment. These approaches, however, rely heavily on image data and image processing techniques, while the majority of them [Adan and Huber, 2011; Budroni and Böhm, 2010b; Díaz-Vilariño et al., 2014] can reconstruct only the doors and not the windows on the walls. In addition, some of them are able to reconstruct the scene only under ideal conditions and lack of occlusions [Budroni and Böhm, 2010b], which restricts their applicability drastically. Older approaches have also tried to detect the wall openings in indoor environments by analyzing the point density and classifying the low-density areas in wall surfaces as wall openings [Hinneburg and Keim, 1998; Ester et al., 1996].

More sophisticated approaches rely on machine learning techniques in order to cluster and model the holes in the walls [Adan and Huber, 2011; Dumitru et al., 2013; Xuehan et al., 2013; Ochmann et al., 2016; Sui et al., 2016], which however increases the algorithmic complexity and computation time, while it requires massive test data for training purposes. Unavoidably, in some cases the segmentation results can be erroneous or biased due to the limited access to adequate training data [Ochmann et al., 2016], while there are cases where user interaction is required for reconstructing successfully the window structures [Sui et al., 2016].

4.1.3 Research Contribution

This chapter introduces a novel method for automatically reconstructing the architectural wall elements of indoor environments, such as windows and doors, under the presence of significant amounts of clutter and occlusion. In contrast to previous methods, our reconstruction pipeline does not require any manual tuning and it does not rely on additional imagery or depth data, avoiding the restrictions and limitations introduced by the previously referenced techniques. Additionally, our method, presented in a conference talk [Michailidis and Pajarola, 2015] and appeared in an extended version as an article in a scientific journal [Michailidis and Pajarola, 2017], is applied to the architectural building models that are reconstructed by any related method in literature, and this constitutes a big advantage of the proposed approach, since it could be embedded as an additional processing module in any of the state-of-the-art modeling pipelines for enhancing and semantically enriching its results.

4.2 Method Overview

To overcome the limitations of current reconstruction approaches, we propose a pipeline that formulates the wall elements detection problem as a graph-cut optimization problem on a wall's 2D cell complex representation. A general diagram presenting the main steps of our reconstruction pipeline is shown in Figure 4.2.

As already mentioned, our method uses as input the architectural 3D model of the indoor environment, which can be typically reconstructed by acquiring the scene using a LiDAR device and then segmenting the raw 3D point cloud into its main building elements, such as the walls, ceiling and floor. Up to this stage, our approach is invariant to the reconstruction method used, thus any modeling pipeline that can reconstruct the general 3D shape of building interiors (e.g. [Okorn et al., 2010; Kotthauser et al., 2013; Mura et al., 2014; Oesau et al., 2014]) could be used for providing the planar surface models to our method. In our work, we used the method described in [Mura et al., 2014].

Before starting the reconstruction of the wall elements, our method assumes, similar to the above-mentioned modeling pipelines, that the buildings' interior structures can be approximated by piecewise planar primitives, an assumption that generally holds true for the vast majority of scenarios [Oesau et al., 2014; Volk et al., 2014]. Then, it extracts from the architectural 3D model the planar surfaces that correspond to the structural building elements (walls, ceiling, floor), selects the wall surfaces and computes their outline shapes using an α-shapes algorithm. The computed outline polygons constitute a higher order representation of the wall surface features, in which we apply a robust multi-line model fitting algorithm in order to get its regularized boundaries. We then reduce the line model space by a robust clustering technique, where the best candidate line models are identified and selected from each cluster as the representative lines of the clusters in the wall surface. In a parallel step, we further enhance the wall surface representation by using an occlusion detection approach, in order to identify and label the occluded points in a voxelized grid of the scene.

In the last stage of our pipeline, we exploit the previously extracted information, in order to segment the wall surface and detect the wall elements. Initially, we form a 2D cell complex structure from the intersection points between the representative line models and we formulate the wall element segmentation as a global optimization problem. As such, we represent the 2D cells of the cell complex by means of an adjacency graph, which is partitioned using a max-flow/min-cut graph-cut approach. The graph-cut optimization segments the cells into regions belonging either to the solid wall, or to areas that represent the open parts, i.e. windows and doors. Extending the generated 2D cell decomposition to the boundary edges of the wall surface, we create a semantic representation of the segmented regions, identifying the solid wall and the wall openings.

4.3 Features Computation

4.3.1 Wall Outline

After extracting the wall surfaces from the input architectural 3D model, the first step in our reconstruction pipeline is to compute the interior and exterior boundary outlines of each wall surface. In our method, each 3D representation of a wall surface W consists of a set of points $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \dots \mathbf{p}_{N-1}\}$ and is defined in our modeling process by its plane normal vector $\hat{n}$, some point $\mathbf{q} \in \mathcal{P}$, the wall's oriented bounding box (OBB), as well as the wall outline polygon from the intersection of OBB with the wall plane.

Since this wall surface representation is quite rough, it needs to be enriched with higher level details. In order to enhance it and provide a description of its outline shape, we compute in the wall plane $(\hat{n}, \mathbf{q})$ the α-shape of its points $\mathcal{P}$ (see

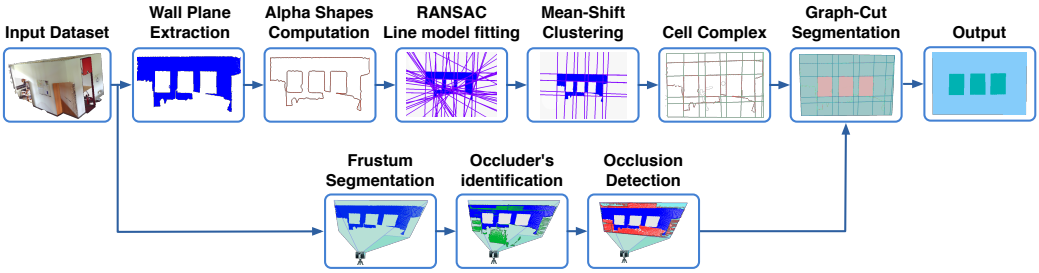


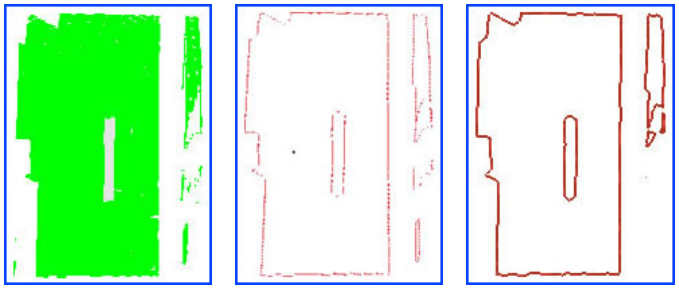
Figure 4.2: *Block diagram of the proposed wall openings reconstruction technique.*

Figure 4.3) [Edelsbrunner and Mucke, 1994]. An $alpha$-shape is a polytope which can be either concave or convex, and is derived from the Delaunay triangulation of $\mathcal{P}$ [Delaunay, 1934]. As such, the α-shape is a sub-graph of the Delaunay graph and can be considered as a generalized hull of a given point set, with the level of detail of the resulting shape dependent on a value α. This user specified parameter α (radius) determines the complexity and smoothness of the computed boundary, and in our approach was chosen conservatively, such as to be sufficiently large compared to the inverse of the scan point density. We should notice also that, in order to simplify the 3D representation of a wall surface and boost the computation process of its α-shape, we project first the wall points onto a 2D plane, reducing the dimensionality of the problem and allowing faster computations.

Despite the robustness of α-shapes for shape reconstruction, there are cases where their boundaries may be rough or contain cluttered segments, while they can be also affected by the introduced bias at the borders [Wei et al., 2007]. Therefore, we refine the computed α-shape boundary of the wall surface by establishing additional evaluation criteria and the α-shapes that do not fulfill them are filtered out. Specifically, we set the minimum polygon surface area to $15cm^2$, assuming that any room window or door is bigger than this size. This value was carefully selected after extensively examining the wall openings of a high number of indoor environments, where none of them had a size smaller than that. In addition, all open polylines and isolated α-extreme points (i.e. outliers) are also detected and filtered out, assuming that they cannot form the desired wall element shapes. One such an example is shown in Figure 4.3.

4.3.2 Line Model Fitting and Clustering

The α-shapes boundaries described above only produce a rough estimation of the true underlying object boundaries and therefore, in order to get the regularized boundaries of the wall surface and wall openings, we apply a line model fitting algorithm to robustly fit straight line segments to the computed α-shapes



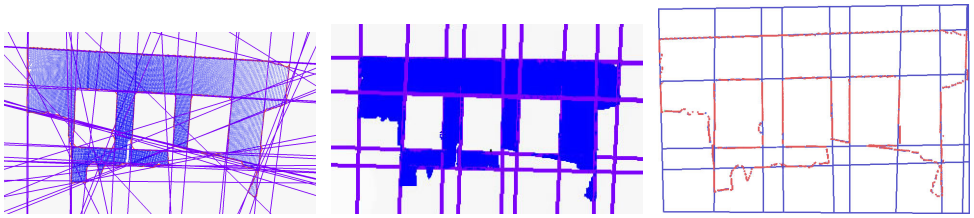
(a) (b) (c)

Figure 4.3: *(a) Wall surface (green) points representing a door. (b) α-shape boundary points. (c) Refined α-shape points.*

boundaries. A good candidate solution for this task is the RANSAC model fitting algorithm, which has been used also in other methods for similar tasks, e.g. in [Kotthauser et al., 2013]. Thus, applying RANSAC to the detected α-shape point set $\mathcal{A}$, we compute the set $\mathcal{Q} = \{\mathcal{Q}^0, \mathcal{Q}^1, \ldots \mathcal{Q}^{l-1}\}$ of the best l line models derived from $\mathcal{A}$ (Figure 4.4(a)). In order to favor the alignment with the wall elements and reduce the complexity of wall surface partitioning, we further reduce the line space $\mathcal{Q}$ to a set of mode line models $\mathcal{M} = \{\mathcal{M}^0, \mathcal{M}^1, ..., \mathcal{M}^{m-1}\}$ using mean-shift clustering [Comaniciu and Meer, 2002]. This approach has also been employed in a similar manner by other state-of-the-art approaches [Furukawa et al., 2009; Oesau et al., 2014], aiming at merging almost co-linear line segments into a reduced set of representative lines that describe the main orientations of the walls in the environment. Thus, in our setup, assuming that the wall's boundary lines are almost parallel to the corresponding polygonal line segments of windows and doors, each line model computed by RANSAC in $\mathcal{Q}$ is assigned to one of the modes in $\mathcal{M}$ based on the following process. First, a clustering is performed only on the orientations of the lines, extracting the major line directions. Then, for each representative orientation, a 1D mean-shift clustering finds the most likely line offsets of that orientation. Thus, the resulting set of line clusters $\mathcal{C}$ correspond to a partitioning of the line models $\mathcal{Q}$ and each cluster in $\mathcal{C}$ defines a representative mode line from $\mathcal{M}$ (see also Figure 4.4(b)).

4.3.3 Cell Complex Creation

The clustered mode lines in $\mathcal{M}$ induce a partition of the wall surface from which the 3D model of the wall can be derived. This data structure of line segments constitutes a 2D cell complex or an arrangement of lines [Edelsbrunner et al., 1986], where each edge of the complex is associated to a representative mode line



(a) (b) (c)

Figure 4.4: *Cell complex construction. (a) The detected wall surface and the line models fitted to the (red) α-shapes outline. (b) The representative mode lines after mean-shift clustering. (c) The 2D cell complex $\mathcal{I}$ (blue) imposed by the representative mode lines on the wall outline polygon. Here, each 2D quadrilateral constitutes a separate cell.*

from $\mathcal{M}$. Using a set of infinite lines $\mathcal{I}$ for representing the mode lines in the cell complex, the cell complex is defined by all intersections of these lines in the wall plane and is bounded by the wall's outline polygon, as shown in Figure 4.4(c).

Additionally, in our cell complex structure, its edges are explicitly associated with the α-extreme points from $\mathcal{A}$ in a way that will allow us later to use this information for segmenting the wall surface. To do so, we associate all α-extreme points of the model lines in a cluster with the mode line of that cluster by projecting them onto it. In this way, the number of α-extreme points associated with each segment in $\mathcal{I}$ will be used in our graph-cut implementation as a contribution factor to the estimation of the edge weights (see also Section 4.4.3).

4.3.4 Occluded Regions

Once all mode lines are detected, the occluded areas and the actual empty space in the wall surface have to be determined, in order to accurately identify the positions of real wall openings. This step is important because, opposite to the facade reconstruction applications, where the windows are detected as holes in the point cloud, in indoor environments this assumption does not generally hold. Due to the large variety of human objects inside the rooms, occlusions and clutter significantly affect the outcome of the wall surface reconstruction, since they produce holes which have to be distinguished from real wall openings.

Thus, it is necessary to accomplish an occupancy analysis to the wall surface in 3D space, in order to identify the different types of holes in the wall surface and distinguish the occluded from the non-occluded regions. The first step in this process is to discretize the 3D space into a uniformly spaced data structure, known as voxel space. This voxelization process is similar to other state-of-the-art methods [Baler and Allen, 2006; Blaer and Allen, 2007; Adan and Huber, 2011]

and could be described as a downsampling process, where the real 3D points are assigned to tiny 3D boxes in R^3, which comprise the voxel grid. However, unlike the other methods, in our approach all points presented in each voxel are approximated with their centroid instead of the center of the voxel, because in this way the underlying surfaces are more accurately represented.

Next, our occlusion detection mechanism classifies the grid cells of the wall surface according to their occupancy state into three categories: *empty, occluded* and *occupied*. The classification incurs using a ray-tracing algorithm, where a tracing line starting from the laser scanner measures the distance to a grid element inside the scanner frustum, which is formed by the scanner sensor position and the convex wall polygon corners (see Figure 4.5(a)). For practical reasons, we have also defined a maximum distance for setting a limit to the scanning area behind the wall plane.

Following this ray-tracing approach, each voxel inside the scanner frustum is classified into the three aforementioned categories as follows.

- Voxels that contain at least one point, force the ray to stop and are assigned as occupied. For example, the voxels that lie on the wall plane and include points from the wall surface.
- Voxels that allow the ray to pass through them without stopping are assigned as empty. Also, voxels that are between the scanner position and an occupied voxel are also labeled as empty.
- Occluded voxels occur in the wall surface, if the ray stops in an occupied voxel before reaching the wall plane (Figure 4.5(b)).

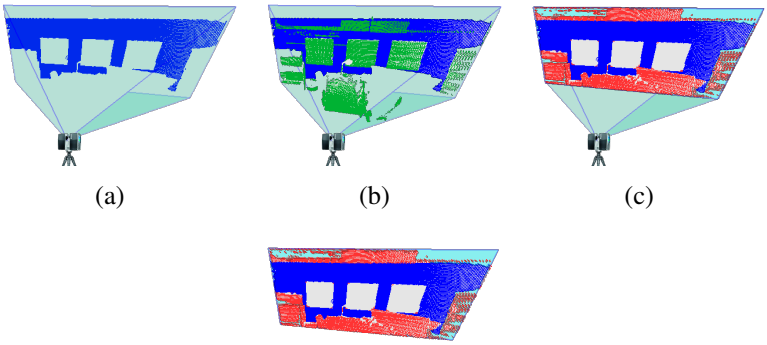
The occluded areas in the wall surface cause many arbitrary holes, which do not correspond to real wall openings and could mislead the wall segmentation process. Thus, we fill these holes with points, which occur by the intersection of the ray that starts from the laser scanner and passes through the occluders, with the wall plane (see Figure 4.5(c)).

This simple yet robust and accurate approach has been proved to be quite good in practice for the reconstruction of the occluded areas, while the information derived from it constitutes an additional decision making factor in the wall segmentation process, as it is described in the next section.

4.4 Graph-based Wall Segmentation

4.4.1 Problem Formulation

The previous steps of the pipeline yield a 2D cell complex that contains many cells which are either filled with points or they are completely empty. In this step,



(a) (b) (c)

(d)

Figure 4.5: *Occlusion detection performed to a wall plane which is formed by the frustum of the laser scanner (wall points are colored with blue) (a). Applying ray-casting, the occluders are identified (with green) (b) and the recovered occluded areas are reconstructed (with red color) (c). After detecting occlusion, the extracted wall surface is enriched with the occluded (red) points (d).*

we need to segment the cells of the complex and assign to them a label, being either wall openings or parts of the wall surface. However, this segmentation is not straightforward and cannot rely only on the number of points the cells contain, due to the inherent uncertainties introduced in the segmentation process. One such example is depicted in Figure 4.6, where different types of uncertain regions are depicted with different colors. All these regions do not constitute wall openings, although some of them could be easily classified as such, since they contain zero or only a few points.

These uncertainties increase the complexity of the wall openings reconstruction problem and, therefore, we need to employ a method to identify which cells of the complex correspond to the wall openings by reasoning on the relations between adjacent cells in the cell complex.

Thus, given the 2D cell complex, we encode the adjacency relationships between the cells using a weighted undirected graph G, in which the nodes represent the cells of the complex and the edges connect spatially adjacent cells. More formally, we associate each cell c_i within the cell complex structure $\mathcal{I}$ to a vertex $v \in V$ in G and tag each vertex by a semantic label from the label set $\mathcal{L} = \{\mathcal{L}_{emp}, \mathcal{L}_{occ}, \mathcal{L}_{unc}\}$ for empty, occupied and uncertain cells, respectively. The graph contains a set of nodes (or vertices) V and a set E of edges $e(u, v)$ with non-negative weights (capacities), which connect adjacent nodes in a regular grid-like fashion and are called n-links. The node set includes also two additional terminal

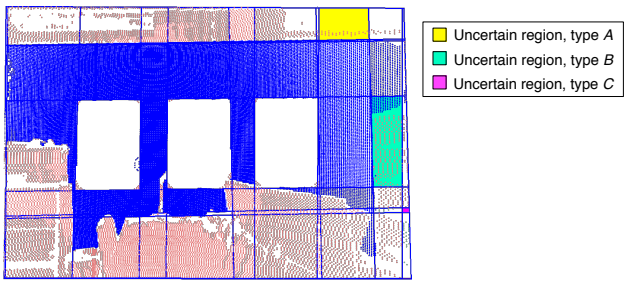


Figure 4.6: *Different types of uncertain regions after occlusion detection (wall surface points and recovered occluded points are depicted with blue and pink colors, respectively). Type A: cells covered only by a few points recovered in the occlusion detection stage and without containing points from the wall surface. Type B: cells with both wall surface points and recovered occluded points in small amounts. Type C: empty cells, which do not include points, although they are not wall openings. Also notice that all three wall openings introduce some additional uncertainties, since they are not fully empty but they contain some points, especially near their corners.*

nodes, the source s and the sink t, and the corresponding edges connecting the nodes with the two terminals are called t-links.

Then, given some energy function and using a minimum graph cut approach on this enhanced dual graph of the cell complex, we can find the optimal labeling of the cells by dividing the nodes V into two disjoint sets S and $T = V - S$, such that $s \in S$ and $t \in T$. The minimum cut in this case can be defined as a subset of edges $C \subset E$ such that their terminal nodes are completely separated on the induced graph $G(C) = \langle V, E \setminus C \rangle$. This is equivalent to finding the maximum flow from s to t.

This segmentation problem can be modeled naturally as a multi-label Markov Random Field (MRF) and can be formulated with unary and pairwise cliques [Boykov et al., 1998]. In this case, the labeling problem can be reduced to the minimization of the following objective (cost) function [Boykov and Funka-Lea, 2006]

$$E(f) = \sum_{p \in V} \mathcal{D}_p(f_p) + \sum_{\{p,q\} \in N_G} V_{p,q}(f_p, f_q) \cdot T(f_p \neq f_q), \tag{4.1}$$

where $f = \{f_0, \ldots f_{|n-1|}\}$ is a binary vector of assigned labels from the label set $\mathcal{L}$ defining the segmentation. Also, $\mathcal{D}_p(f_p)$ is the unary data term of assigning label f_p to node p based on a chosen probabilistic model, while $V_{p,q}(f_p, f_q)$ is the smoothness penalty term of defining the discontinuity cost between two adjacent nodes p and q when their labels are f_p and f_q, respectively, in a specified neighborhood system N_G. The comparison predicate $T(\cdot)$ in Equation 4.1 is 1 if its argument is true and 0 otherwise. In our method, the neighboring system N_G was

a 4-connected lattice of cells that follows the von Neumann adjacency

$$N_G(x_i, y_i) = \{(x, y) :\mid x - x_i \mid + \mid y - y_i \mid \leq r\}, \tag{4.2}$$

assuming that each cell c_i is given by its position indices (x_i, y_i) in $\mathcal{I}$ and r is the von Neumann neighborhood range, which was set to 1 in our approach.

After formulating the segmentation labeling problem, we can compute efficiently the minimization of Eq. 4.1 with graph cuts, assuming that $V_{p,q}(f_p, f_q)$ is a submodular function [Vicente et al., 2008]. Therefore, we assign weights to graph's edges so that the min-cut cost $|C|$ to be equal to the minimum energy $E(f)$. As such, the two energy terms in Eq. 4.1 will represent the two types of edges in the graph, i.e. the t-links and n-links.

4.4.2 Unary Data Term

In our framework, the unary data term $\mathcal{D}_p(f_p)$ is defined using a Bayesian probabilistic model, introducing some spatial priors about the initial state of cells in $\mathcal{I}$. Specifically, we represent the wall cell complex $\mathcal{I}$ by a vector of $|N_c|$ elements, $\hat{\mathbf{z}} = [z_0, \ldots, z_{|N_c-1|}]$, which indicate the estimated class probabilities $Pr(f_p \mid c_i)$ of a cell c_i to belong to label f_p. Applying Bayes' rule, this probability can be expressed as

$$Pr(f_p \mid c_i) = \frac{Pr(f_p)Pr(c_i \mid f_p)}{Pr(c_i)} \propto Pr(f_p)Pr(c_i \mid f_p), \tag{4.3}$$

where $Pr(c_i)$ is a normalizing constant, which is independent from f_p and can be omitted, while $Pr(f_p)$ and $Pr(c_i \mid f_p)$ are computed using spatial information from the cell complex. Here, the prior probability distribution $Pr(f_p)$ of the state f_p encodes our belief in the likelihood of various cell labels, while $Pr(c_i \mid f_p)$ is the conditional probability of seeing the cell c_i given the particular cell label f_p. In our case, it is based on the cell density function

$$\mathcal{D}_{c_i} = \gamma_D \frac{\mid Cov_{c_i} \mid}{\mathcal{S}_{c_i}} = \gamma_D \frac{\mid \mid \mathcal{P}_{c_i} \mid - \mid \mathcal{A}_{c_i} \mid \mid}{\mathcal{S}_{c_i}}, \tag{4.4}$$

where Cov_{c_i} is equal to the number of points covering the cell c_i ($|\mathcal{P}_{c_i}|$) minus the number of α-extreme points ($|\mathcal{A}_{c_i}|$) that lie in the cell's boundary line segments. $\mathcal{S}_{c_i}$ denotes the cell's area in Equation 4.4 and γ_D is a scaling factor.

Then, we compute the conditional probability $Pr(c_i \mid f_p)$ of an arbitrary cell c_i by

$$Pr(c_i \mid f_p) = K \log(\frac{\mid \mathcal{D}_{c_i} \mid}{N_c}) e^{-|\mathcal{A}_{c_i}|/Cov_{c_i}}, \tag{4.5}$$

where K is a proper normalization factor.

In order to estimate the prior probability $Pr(f_p)$, we use an approach which was inspired by [Frome et al., 2006], and we create a distance function that relies on specific features of cells in $\mathcal{I}$. First, we create the set $\mathcal{E}$, where $\mathcal{E} = \{\epsilon_1, \epsilon_2, ..., \epsilon_{n_\mathcal{E}}\}$ is the set of $n_\mathcal{E}$ labeled cells, called *base* cells, which are actually the cells for which we have a high certainty for their labels. Since there are only three possible labels that can be assigned to the cells of the complex, these base cells were categorized into three different classes. Let $\hat{f}_\epsilon$ denote the feature vector describing the base cell ϵ and let $\hat{f}_{c_i}$ denote the feature vector of an arbitrary cell c_i. Then, we create the vector $\hat{d}_{\epsilon c_i}$, which contains the L_2 norms between the individual features describing ϵ and c_i, i.e. $\hat{d}_{\epsilon c_i} = \| \hat{f}_\epsilon[i] - \hat{f}_{c_i}[i] \|$.

Next, we need to define the features that will be used for describing the cells. High attention should be given to the selection of these features, since they should be expressive enough but also sufficiently simple for fast computation. In our case, it was shown that the density $\mathcal{D}_{c_i}$ as a simple feature is sufficient in order to define the certainty of a cell being a base cell, although other features such as the ratio $|\mathcal{A}_{c_i}|/Cov_{c_i}$ could also give similar results. Thus, the base cells selection was based on $\mathcal{D}_{c_i}$, assuming that the cell with the highest density should belong to the wall surface (label $\mathcal{L}_{occ}$), and equivalently, the cell with the minimum density should belong to a wall opening (label $\mathcal{L}_{emp}$).

Then, we can estimate which cells are likely to have the same label as the associated base cell, using the distance function

$$D_\epsilon(c_i) = \beta_\epsilon \hat{d}_{\epsilon c_i} \quad \text{with} \quad \beta_\epsilon = \frac{1}{\beta_d \mathcal{D}_{c_\epsilon}}, \tag{4.6}$$

where the weight β_ϵ defines the range of cell density in which the cells are likely to have the same label, and β_d is a coefficient defining the deviation of cell's density from the base cell density $\mathcal{D}_{c_\epsilon}$. In our approach, this was fixed to 0.2, meaning that every cell whose density is less than 20% from $\mathcal{D}_{c_\epsilon}$ will be assumed to belong to the same set as the base cell ϵ.

To determine the prior probability $Pr(f_p)$, we first set a decision boundary using the inequality $D_\epsilon(c_i) \leq 1$, where a value of 0 corresponds to perfect similarity. This threshold was selected after considering that in other approaches, such as in [Frome et al., 2006], it gave empirically good results. Hence, the prior probability $Pr(f_p)$ can be given by:

$$Pr(f_p) = \frac{| \{c_i \mid D_\epsilon(c_i) \leq 1\} |}{N_c}, \text{ if } f_p = \{\mathcal{L}_{occ} \vee \mathcal{L}_{emp}\}. \tag{4.7}$$

In order to estimate the prior probability for the uncertain cells, we apply the probability's additivity axiom to the previously computed probabilities, since the set of base cells includes only the labels $\mathcal{L}_{occ}$ and $\mathcal{L}_{emp}$. Thus, $Pr(f_p = \mathcal{L}_{unc})$ can

be computed by:

$$Pr(f_p = \mathcal{L}_{unc}) = 1 - Pr(f_p = \mathcal{L}_{occ}) - Pr(f_p = \mathcal{L}_{emp}) \tag{4.8}$$

4.4.3 Smoothness Term

After defining the conditions that determine the topological connectivity of cells in the cell complex, we need also to set the edge weights that connect the cells, in order to perform the graph-cut segmentation. However, before setting the edge weights, we impose certain regional hard constraints for the segmentation by indicating certain cells that should belong to wall surface or wall openings. Generally, the regional constraints on G are vertices in V for which there is high belief that they are adjacent and immediately connected to a specific terminal node, and therefore are called *seed* nodes.

To define these constraints, i.e. the initial seed nodes, we use two threshold values th_H and th_L for the class probabilities of cells in $\hat{\mathbf{z}}$ and we apply the Equation 4.9 to determine the cells that will be assigned with the corresponding labels.

$$f_p = \begin{cases} \mathcal{L}_{occ} & \text{, if } z_i \geq th_H \\ \mathcal{L}_{emp} & \text{, if } z_i \leq th_L \end{cases} \tag{4.9}$$

This initialization is quite important for the segmentation of the wall plane, since it determines the connection of the cells to the corresponding terminal nodes with a sufficiently large (theoretically infinite) weight, in order that these links will not be selected by the min-cut computation. In our approach, the values of th_H and th_L were set empirically to 0.9 and 0.2, respectively.

Next, we compute the edge weights of t-links, which connect the non-seed vertices to the terminal nodes and are computed proportionally to the distance from the corresponding thresholds. For the source terminal link, the edge weight is given by

$$w_p^{(s)} = \frac{z_i - th_L}{th_H - th_L} \cdot \kappa_s, \tag{4.10}$$

where κ_s is a sufficiently large constant which ensures a feasible flow in the graph. In a similar manner, the edge weight for the sink terminal link is defined as

$$w_p^{(t)} = \frac{th_H - z_i}{th_H - th_L} \cdot \kappa_s. \tag{4.11}$$

Moreover, the edge weights for the n-links are set to

$$w_{\{p,q\}} = \begin{cases} \kappa_n(\mathcal{R}_{dens} + \mathcal{R}_a) & \text{, if } p \neq q \\ 0 & \text{, if } p = q \end{cases} \tag{4.12}$$

where $\mathcal{R}_{dens}$ and $\mathcal{R}_a$ are the contributions of certain cell features to the computation of the edge weights, as they are described below. κ_n is a properly selected constant, which ensures that the n-link weights will be in the same range as the non-seed t-links.

The terms in Equation 4.12 are used to penalize the connectivity of adjacent cells that exhibit different features and potentially belong to different segments in the cell complex. The first term $\mathcal{R}_{dens}$ is used to measure the difference of cell densities and is defined as:

$$\mathcal{R}_{dens\{p,q\}} = e^{-\frac{(\mathcal{D}_{c_p} - \mathcal{D}_{c_q})^2}{2\sigma^2}} \tag{4.13}$$

The term $\mathcal{R}_a$ computes the difference of α-extreme points that lie in cell's boundaries, considering that cells with many α-extreme points are highly possible to be near the wall boundaries, while cells with no α-extreme points is highly possible to be inside a uniform region and not near the boundaries. Thus, this feature is a strong indicator of cells that potentially lie near to a transition region, i.e. between two adjacent cells that might belong to the wall surface and a wall opening, and is defined as:

$$\mathcal{R}_{a\{p,q\}} = \mid |\mathcal{A}_{c_p}|^2 - |\mathcal{A}_{c_q}|^2 \mid \tag{4.14}$$

4.4.4 Wall Semantic Segmentation

After assigning the weights to the source and sink t-links, as well as the neighboring node n-links, we can map this problem to a graph-cut problem with its min-cut cost $|C|$ equalling $E(f)$, using the following edge weights assignments [Boykov and Funka-Lea, 2006]:

$$\begin{aligned} \mathcal{D}_p(f_p) &= w_p \\ V_{p,q}(f_p, f_q) &= w_{\{p,q\}} \end{aligned} \tag{4.15}$$

Since all the pairwise potentials used satisfy by definition the submodularity property, the exact solution of the global minimum cut with only two terminal nodes can be computed by minimizing the energy function of Equation 4.1 in polynomial time [Boykov and Funka-Lea, 2006]. The minimum energy is associated to the optimal labeling of the cells, while the wall elements of the segmented wall surface can be reconstructed by unifying the cells with the same label.

Finally, to semantically annotate the reconstructed wall openings, we can adopt a simple heuristic approach, where each wall element is evaluated against certain shape and positional criteria in order to get the proper label (e.g. window or door). For example, wall openings attached to the lower boundary of the wall surface could be labeled as doors, while openings surrounded by the wall surface could be labeled as windows.

Table 4.1: *Datasets statistics, showing the total number of points in the point cloud (*#Tot. Points*), the number of points in the selected wall plane (*#Pnts/Wall*), the number of scans (*#Scans*), the number of wall openings (*#WO (GroundTruth)*) and their semantic annotation (*'Semantics'*).*

Dataset	#Tot. Points	#Scans	#Pnts/Wall	#WO (GroundTruth)	Semantics
OFFICE1	0.63m	1	24629	3	Windows
OFFICE2	2.68m	1	131704	1	Door
OFFICE3	2.68m	1	16456	3	Windows
OFFICE4	8.13m	3	256817	1	Door
OFFICE5	0.63m	1	3747	8	Windows

4.5 Results

The effectiveness of our reconstruction and modeling pipeline was quantitatively and qualitatively evaluated on a test suite composed of various datasets from different real-world indoor environments. Indicatively, we show in this section the results from five real-world datasets of building interiors, which were acquired by [Mura et al., 2014] and present different wall surfaces and occlusion levels. We focused mainly on datasets from office buildings, which typically present more complex window frames than those in domestic buildings, and thus, they can challenge more our reconstruction pipeline.

Although our pipeline processes and segments automatically all wall surfaces of the input environments, we will focus our discussion in this section only on the wall surfaces of the test datasets that present different showcases of wall openings and challenge more our method, while we will omit the less interesting ones or the ones that do not contain wall openings. Note also that, the selected wall surfaces in our test bench include windows and doors, and our algorithm was evaluated in both cases without manual tuning, as opposed to other techniques in the literature, such as in [Díaz-Vilariño et al., 2014; Zhang and Zakhor, 2014], which are particularly tuned to identify only one of them. All the relevant statistics about the datasets we used and their processing times are provided in Tables 4.1 and 4.2, while Figures 4.7, 4.8 and 4.9 present an overview of the point clouds we used, the selected wall surfaces and the segmentation results.

Implementation For the prototype implementation of our approach we used C++ and some publicly available libraries, such as the CGAL [CGA, 2018] and VTK [VTK, 2018], while the datasets of our test suite were captured by a high-end LiDAR range scanner. All tests were performed on a Mac Pro with a Quad-Core

Table 4.2: *Processing times for each dataset and for each stage of our pipeline, i.e. for Alpha Shapes computation (*AS*), line model fitting (*LM*), occlusion detection (*OC*), cell complex construction and graph-cut segmentation (*CC+GC*).*

Dataset	AS (sec)	LM (sec)	OC (sec)	CC+GC (sec)	Total (sec)
OFFICE1	1.44	0.15	0.37	17.81	19.76
OFFICE2	7.54	0.04	1.36	19.78	28.72
OFFICE3	0.85	0.15	1.41	8.12	10.71
OFFICE4	7.39	0.01	4.31	44.12	55.82
OFFICE5	0.18	0.09	1.09	14.17	15.53

Intel Xeon processor (2.8GHz), 16GB DDR2 RAM and an NVIDIA Quadro FX 4800, without using any multithreading technique.

Table 4.2 provides the processing times for the main stages of the proposed reconstruction pipeline. As we notice in the first column of this table (column AS), the highest processing times for the α-shapes computation are presented for the wall surfaces of OFFICE2 and OFFICE4 datasets, which are almost 7 times higher than that of the other wall surfaces. Although from Table 4.1 we can see that these two wall surfaces contain more points than the other wall surfaces, their high processing times cannot be fully justified by the number of points they contain. In particular, although the wall surface from OFFICE4 contains almost twice the number of points from OFFICE2, the processing times for computing the α-shapes for both wall surfaces are similar. Thus, this difference is mainly justified by the fact that, the $alpha$-shapes computation is highly dependent not only on the number of points of the surface under consideration, but also on the complexity of its object boundaries, which affects the number of Delaunay triangulations necessary for the computation of the outline wall polygon. Since the wall surface of OFFICE2 presents more holes than that of OFFICE4 (see also Figure 4.8), its α-shapes computation is more demanding and therefore, an increase in its processing time is reasonable. We also notice from the results in Table 4.2 that the graph-cut segmentation is the most time-consuming part of our pipeline, while the overall processing time for OFFICE4 is higher, mainly due to the higher number of points it contains.

Quantitative and qualitative evaluation Table 4.3 presents the statistics from the main reconstruction stages of our pipeline for each wall surface of the datasets we used. It also includes a comparison between the true number of wall openings evident in the wall surfaces and the number of the detected ones from our pipeline. Overall we can see that, our method performed quite well in

Table 4.3: *Processing statistics, showing the number of: line models detected by the line model fitting algorithm (*#LM*), lines which comprise the cell complex (*#CC lines*), occluded points and their percentage in wall surface (*#OccPnts(%)*), Alpha Shapes points before (*#AS*) and after (*#AS(opt.)*) the refinement process, cells in cell complex (*#Cells*), wall openings in ground truth and detected by our method (*#WO (GT/Det.)*, as well as their* Semantics(#)*).*

Dataset	#LM	#CC lines	#OccPnts (%)	#AS	#AS(opt.)	#Cells	#WO(GT/Det.)	Semantics(#)
OFFICE1	49	18	10071 (29%)	1367	1365	60	3/3	Windows (3)
OFFICE2	25	12	6998 (5%)	2311	2037	25	1/1	Door (1)
OFFICE3	25	16	12214 (43%)	2123	2098	45	3/3	Windows (3)
OFFICE4	8	12	48380 (16%)	1789	1131	25	1/1	Door (1)
OFFICE5	38	32	6924 (65%)	1149	951	224	8/7	Windows (7)

all tests and was capable to detect and segment all wall openings correctly, besides one misclassification error occurred in OFFICE5. This misclassification is however justified by the complexity that this wall surface presents (see also Figure 4.9), where on the lower left part of the wall, the intermediate segment of the window frame between the two small windows is completely missing due to a viewpoint occlusion. Since this part of the window frame was very thin, it could not be fully recovered and hence the two windows were merged in the graph-cut segmentation stage (see Figure 4.9(f)).

Figure 4.7 presents the results of the main pipeline stages of our method for the wall surface of OFFICE1. For convenience, in this dataset we have marked the points of individual wall planes with different colors, where the wall of interest was colored with blue. A general overview of the room interiors is presented in Figure 4.7(a), while Figure 4.7(f) shows the final result of the segmented wall surface, after the application of the graph-cut algorithm.

Figure 4.8 shows the segmentation results from three more office environments. Each of these datasets presents specific difficulties that challenge the segmentation capabilities of our method. Specifically, the OFFICE2 dataset (Figure 4.8(a)) contains an open doorway and a wardrobe in front of the wall, while there are many other objects that create additional occlusions in the wall surfaces. Although the wardrobe hides a big rectangular part on the right side of the extracted wall surface, which could be wrongly interpreted as a doorway, the extracted results (right image in Figure 4.8(a)) indicate that our method was capable to successfully detect the doorway and classify this large gap as part of the wall.

OFFICE3 is another challenging dataset (see Figure 4.8(b)) which contains a big window with three adjacent frames that cover the most part of the wall. Also,

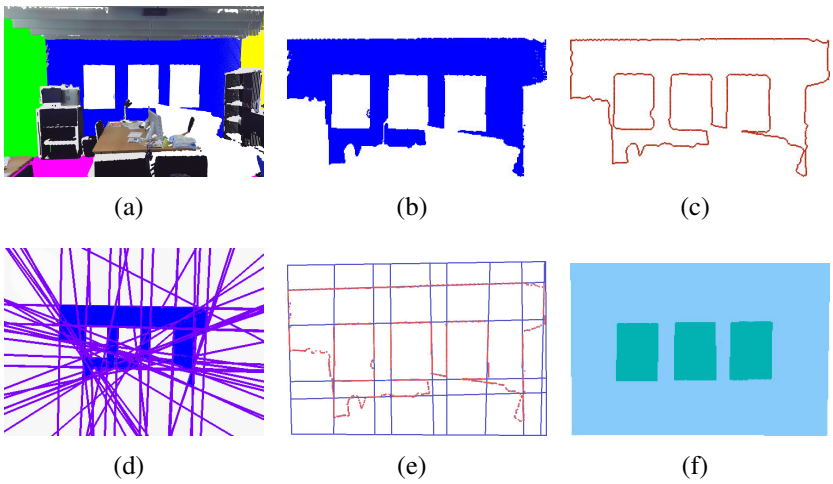
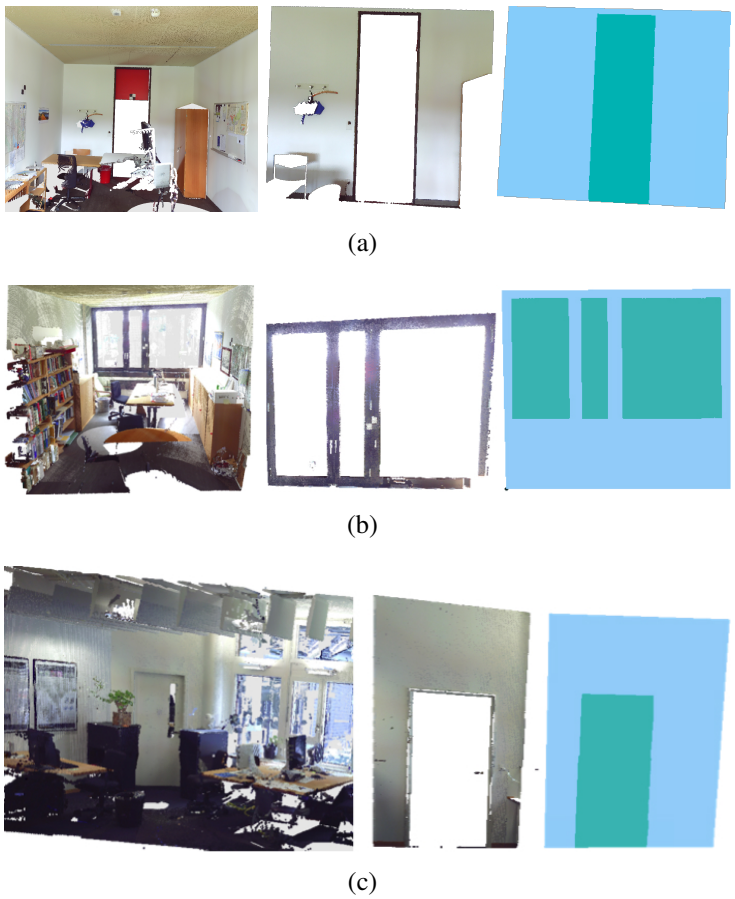


Figure 4.7: *Results from the main pipeline stages for OFFICE1. (a) The room environment with colored wall surfaces, (b) the selected wall surface, (c) the Alpha Shapes, (d) the line models, (e) the cell complex and (f) the graph-cut result.*

it contains many interior objects (occluders), which hide a large part of the remaining wall surface and parts of the window frames too, introducing additional difficulties to the reconstruction process. Nevertheless, our algorithm was able to correctly identify all wall openings, reconstructing also the missing window part. Notice here that many recent approaches, e.g. [Ochmann et al., 2016], are not capable to reconstruct accurately the wall openings in such a common scenario, since they cannot handle wall surfaces which contain multiple adjacent wall openings.

A different scenario with a closed door and some closets in front of it that cover part of it but also part from the door frame, is examined in OFFICE4 dataset (see Figure 4.8(c)). Also in this case, despite the challenging room environment, the doorway was accurately detected by our pipeline after the graph-cut segmentation. Note here that this showcase reveals an important advantage of the proposed method against other methods in literature, where under certain circumstances our approach can detect the doorways even if the doors are closed, a scenario that the majority of state-of-the-art methods would fail.

In Figure 4.9 we show the segmentation results from the most challenging dataset of our test bench. The examined indoor environment contains a complex and irregular combination of window frames, which is an exceptionally difficult setting for the laser scanner. The extracted wall is highly cluttered and occluded, where up to 65% of its wall surface was completely occluded, causing many window frame segments to be corrupted or missing. Therefore, a high number of



(a)

(b)

(c)

Figure 4.8: *Wall surface reconstruction for (a) OFFICE2, (b) OFFICE3 and (c) OFFICE4. For each dataset, the input point cloud (left), the extracted wall plane (middle) and the wall openings segmentation (right) are presented.*

cells is generated after the cell complex creation (Figure 4.9(e)) and the accurate segmentation of the correct cells becomes a very difficult task. Despite these challenges, our algorithm managed to detect correctly almost all window frames, although as described previously, due to its complexity some cells from the cell complex were erroneously interpreted and led to misclassifications.

4.6 Conclusions and Discussion

In this chapter we presented a generic fully automatic 3D modeling pipeline, which goes beyond the current reconstruction techniques and is capable to reconstruct accurately the architectural wall elements of indoor environments under

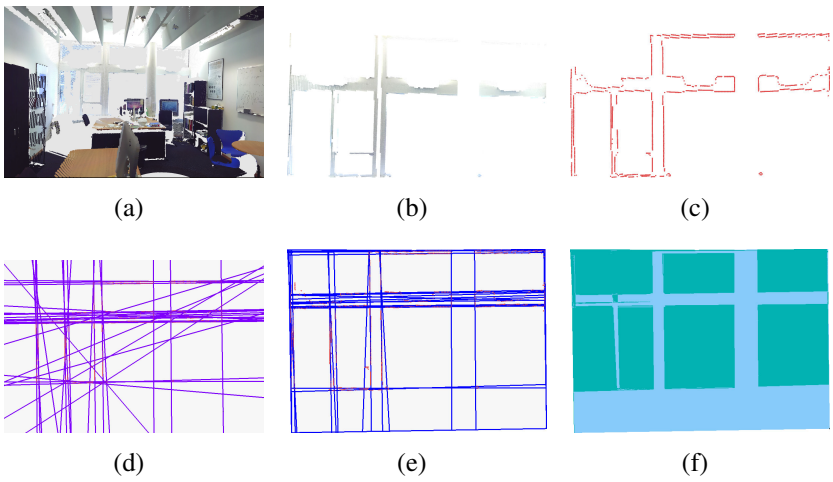


Figure 4.9: *Results from the main pipeline stages for OFFICE5. (a) The room environment, (b) the selected wall surface, (c) the Alpha Shapes, (d) the line models, (e) the cell complex and (f) the graph-cut result.*

very challenging conditions. It emphasizes in the segmentation and modeling of the wall elements in complex indoor environments without human intervention, where their reconstruction until now was not possible. The segmentation was approached by partitioning each wall segment to a collection of smaller regions connected together according to a certain topology, whose topological and spatial relationships were exploited by a suitable graph representation.

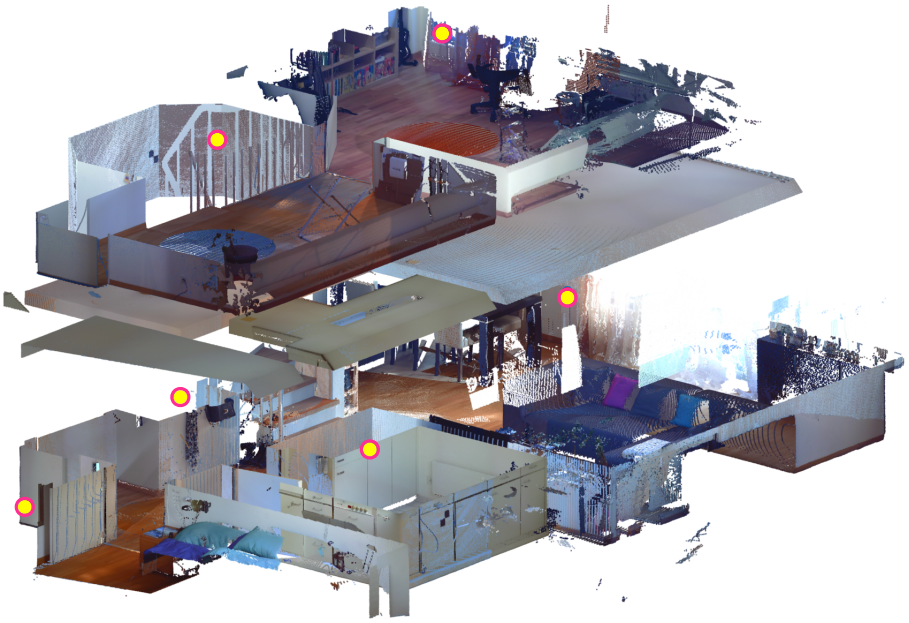
The evaluation process we performed demonstrates the efficiency of the proposed wall elements reconstruction pipeline on a variety of challenging real-world scenarios and proves that our method can work very well in practice. However, despite its robustness, the proposed approach presents also some limitations, which were superficially mentioned in Section 4.5. For instance, although our pipeline works quite adequately in many real-world scenarios, there are some cases (such as the wall surface in Figure 4.9) where the absence of some window frame segments can lead to false classifications. Additionally, the over-segmentation of the wall plane by the model lines can create in some cases a huge number of cells with various shapes (triangles and rectangles), which might mislead the graph-cut optimization process and lead to misclassifications.

Nevertheless, there are still various aspects that could be further improved in our approach. Among others, a pre-processing step applied to the cell complex structure could enforce the shape consistency in cells, merging together small adjacent cells, in order to form quadrilateral cells before the graph-cut optimization step, allowing a more reliable and accurate segmentation. An even more interest-

ing extension would be to adjust the pipeline, in order to be capable to reconstruct non-rectangular wall openings, e.g. windows with curved frames.

CHAPTER 5

SCANNER POSITION RECONSTRUCTION



5.1 Introduction

In the previous chapters, we introduced some methods that are capable to improve specific parts of the reconstruction pipeline and refine their extracted results. This chapter discusses a different yet novel pipeline that retrieves a vital information necessary to almost all reconstruction and modeling methods in the literature. Specifically, our approach recovers the original scanner viewpoint positions from the raw point clouds, an information used from modern methods in various applications, such as for the reconstruction of the architectural 3D model of the indoor environment, the detection of occlusions, the reconstruction of the wall elements or the semantic partitioning of the room spaces of the building.

Our method can automatically compute the original scanner position coordinates exploiting only the information derived from the raw point data and is mainly applicable to merged (i.e. registered) point clouds, where the original scan positions are typically lost and not available. Typical applications and tasks of the general reconstruction pipeline that could mainly benefit from our method are annotated in Figure 1.2 with orange color and refer to the third contribution of this thesis.

5.1.1 Motivation

In recent years, many methods have been proposed in the field of building reconstruction that cover almost all aspects of the modeling pipeline, e.g. they can reconstruct the architectural structures of indoor environments (e.g. walls, ceiling and floor), can detect the wall openings, can successfully handle occlusions and clutter, while they can also semantically subdivide the building space into individual rooms.

Due to the diversity of the applications in the field and the increased complexity of the reconstruction problem, the target of all these methods is to be sufficiently effective and generic to support the extraction of higher-level architectural models from the raw point-cloud data. This is mainly achieved by following a reconstruction and modeling process similar to the one outlined in Chapter 2 (and illustrated in Figure 1.2). Despite, however, the capabilities of all these methods to generate automatically faithful architectural 3D models, the majority of them rely heavily on the prior knowledge of the scanner viewpoint positions in order to be executed successfully. However, this information gets typically lost after registering the raw point clouds and is not available to the next stages of the modeling process. Thus, without knowing these positions a priori, none of these methods could work and their modeling pipelines would have failed, while in this sense, none of them could be characterized as automatic due to their dependency to this prior, which requires human intervention in order to be recovered again.

Surprisingly enough, the current literature does not offer yet a solution to this problem and no other method has been proposed so far for re-engineering and reconstructing this valuable information. It would therefore be beneficial, if there would be a method capable to automatically reconstruct the original scanner positions, allowing the current methods to lift this restrictive assumption and perform the reconstruction tasks without requiring human intervention or imposing such hard constraints.

5.1.2 State of Research

To better comprehend the extent to which the scanner viewpoint positions are used in state-of-the-art reconstruction methods for generating the architectural 3D model of the indoor environment, in this section we provide a brief overview of the related literature, emphasizing mainly on the usage and exploitation of this information.

Many traditional approaches [Adan and Huber, 2011; Xiong et al., 2013; Previtali et al., 2014; Mura et al., 2014] reconstruct the 3D model of the environment by clustering first the raw point clouds into planar patches, which are then classified as floor, ceiling, and walls, considering at the same time clutter and occlusions. Despite their efficiency, methods in [Adan and Huber, 2011; Xiong et al., 2013] rely on the prior knowledge of scanner positions in order to perform occlusion labeling, wall openings detection and wall surface reconstruction, while the pipeline in [Previtali et al., 2014] uses the viewpoint information for hierarchically clustering the structural elements and accurately reconstructing the wall openings. The prior knowledge of the number of scans and the scanner positions is also used extensively in [Mura et al., 2014] for extracting the vertical planar patches on a per-scan basis and employing viewpoint-based visibility computations for occlusion detection and wall reconstruction. In addition, the extraction of the correct number of rooms and room layout in the same method relies explicitly on the number of scanner positions, which is assumed to be known a priori.

In another method [Sanchez and Zakhor, 2012], consistent vector orientations are assigned to the computed point-wise normal vectors according to the scanner positions, in order the points of the point cloud to be classified and segmented into floor, ceiling and walls, while similarly in [Stambler and Huber, 2014], the scanner viewpoint information is used for processing and segmenting the source point cloud on a per-scan basis, in order to estimate the orientation of the surface models and detect occlusions. Another approach aiming for a volumetric building model was presented in [Ochmann et al., 2016], where a multi-label minimization framework was solved to segment the input point cloud and generate the room layout relying on the prior that the initial set of room labels has been provided directly by the scanner positions. Additionally, this prior information was used

again in the same method for the detection of the wall openings.

In a recent extension of [Mura et al., 2014] presented in [Mura et al., 2016], a graph-based scene representation approach was used for partitioning the building environment into rooms, using visibility weights and a multi-label Markov Random Field process. In this pipeline, the identification of the correct viewpoint cells and the partition of 3D space relies explicitly on viewpoints, as well as the visibility-based room clustering approach for finding the approximate room locations and creating the final room layout.

Our previous method, presented in [Michailidis and Pajarola, 2015; Michailidis and Pajarola, 2017] and outlined in Chapter 4, relies also on the prior knowledge of scanner positions for detecting occlusions and classifying the cells of the cell complex before the wall surface segmentation. In a more recent pipeline [Ambrus et al., 2017], the input point cloud is projected into 2D in order for the interior building model to be reconstructed using an energy minimization approach. As in previous approaches, this method relies also on the viewpoints, in order to segment the 2D space and reconstruct the room layout, with the only difference that it proposes a technique to compute synthetic viewpoints instead of using the ones provided by the point cloud metadata.

Although the modeling pipelines of these state-of-the-art methods are quite efficient and robust, their results rely explicitly on the prior knowledge of the scanner viewpoint positions, which introduces inevitably a hard constraint to their reconstruction process and restricts their automatic operation.

5.1.3 Research Contribution

The contribution described in this chapter introduces the first method that allows the automatic reconstruction of the real scanner viewpoint positions, without requiring any prior knowledge about the environment or the dataset. Our approach is capable to identify and compute automatically and accurately (with a precision of up to tenths of millimeters) the laser scanner positions in large datasets, exploiting only information derived from the raw point data. This novel pipeline is independent of the architectural shape of building interiors, and therefore, it can be applied to any environment with complex room layouts, arbitrarily oriented planar or curved walls and sloped ceilings, while it can be applied more beneficially to merged point cloud data (e.g. after registration), where the original scan positions are typically lost. Moreover, it is quite robust against outliers, noise and heavily cluttered scenes, while it is not affected by occlusions. In addition, being independent from the laser scanner manufacturers, makes it suitable for almost every real-world LiDAR application, allowing the current reconstruction and modeling pipelines to take advantage of this vital information produced by our method, in order to become independent from expensive manual computations and move one

step forward towards their fully automatic operation.

5.2 Method Overview

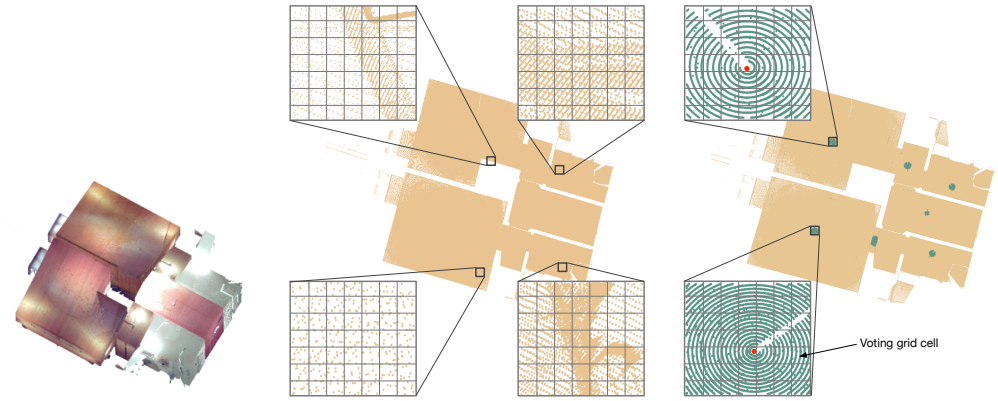
Relying on the intrinsic scanning characteristics of the point cloud acquisition process outlined in Section 2.2.1, our method employs a descriptive, spatial point pattern analysis to the point distribution of the raw point cloud, in order to reveal the features which will allow the retrieval of the original scanner viewpoint positions. Emphasis has been given to the detection of the circular scanning patterns and the high density of points near the scanner positions, since as was described in Section 2.2.1 and illustrated in Figure 2.1, these features can clearly indicate the accurate spatial position of the scanning device.

Being independent of the architectural shape of building interiors, the input to our pipeline can be a set of merged cluttered 3D point clouds representing any kind of interior environment and acquired by any laser scanning device as outlined above (see also Section 2.2). Thus, given the input dataset and following a strategy for reducing the space dimensionality of processing the source point cloud from 3D to 2D for performance reasons (Figure 5.1(a)), our method computes its bounding box and generates a uniform 2D grid of cells, into which the 3D points of the input point cloud are projected and classified (Figure 5.1(b)). Then, we perform spatial and point pattern analysis on the point distribution of each cell, creating a cell feature vector which is evaluated by a linear *support vector machine* (SVM) for identifying strongly reliable *voting cells*, which are cells that indicate with high confidence the scanner positions in the grid (Figure 5.1(c)).

After detecting the voting cells, we cluster their points by applying *Gaussian mixture models* (GMM) in order to detect any spatial gradual variations in point pairwise distances and point densities. Taking also into account the curved-like shape of the spatial point distribution of voting cells (see also the insets in Figure 5.1(c)), a circle model fitting algorithm is applied to each point cluster of a voting cell, estimating its circle center. The resulting circle centers are next fed into a Hough voting space, where after applying non-maximum suppression, the maxima of votes are identified, indicating the cells where the scanner positions lie in the grid. By further analyzing the spatial distribution of points inside these selected cells and after applying mean-shift clustering, our approach finds the exact position of the scanner devices within an accuracy of tenths of millimeters.

5.3 Detection of Voting Cells

The first step in our processing pipeline is to project the input point cloud into a 2D grid of cells, and detect the areas (i.e. cells) where the point distributions



(a) Input point cloud.

(b) Grid plane of the input point cloud, where the 3D points were projected and classified into the cell grids.

(c) Detected voting cells, which will cast votes for the scanner positions.

Figure 5.1: *The main stages for the reconstruction of scanner positions. The input point cloud (a) is first projected onto a 2D plane and its points are classified into a 2D grid of cells (b). Then, a SVM classifier detects the reliable voting cells which will be used for computing the scanner positions (c). The image insets in (b) and (c) indicate the clear difference of point distributions between randomly selected cell areas in the projected point cloud and the cells near and around the scanner positions. The finally retrieved scanner positions are indicated with a red dot inside the insets in (c).*

clearly indicate the circular scanning patterns of LiDAR devices. These cells are denoted in our method as reliable voting cells, since in a later step they will cast votes for the potential scanner positions.

Before proceeding to the detailed description of our approach, it would be an omission to ignore mentioning some of its advantageous characteristics. Firstly, since our method aims at processing large-scale datasets, a point-based analysis of the whole dataset in 3D space would be very time-consuming and computationally expensive. Thus, we developed a strategy for processing the raw point cloud in 2D rather than in 3D space, reducing the dimensionality (and hence the complexity) of the problem and boosting the processing efficiency for large datasets. A further discretization of the continuous 2D space into a grid of uniform cells helped to efficiently balance between the computational performance of the method and the position accuracy for the localization of scanner devices. Moreover, applying advanced statistical methods and analyzing the spatial point distribution of each cell, we created a descriptive statistical feature vector that allows the reliable classification and detection of the voting cells.

5.3.1 Space Partitioning

To create the 2D cell grid, in which the points of the point cloud will be projected, we compute first the oriented bounding box (BB) of the input point cloud, which provides a simplified shape approximation of the scene and allows us to demarcate the equivalent 2D space into which the points will be projected. Based on this bounded 2D space generated by the BB, a uniformly subdivided grid of cells is created, which all 3D points are projected into and assigned to the respective cells.

In this construction, attention should be paid to the definition of the size of grid cells, which controls the density of the grid and consequently the number of embodied points of cells, which intuitively will determine also the level of expressiveness of the features derived from them. Thus, we need to consider, not only the intrinsic characteristics of the acquisition process as described in Section 2.2.1, but also the scanning point patterns generated by the LiDAR device and the impact of cell size to the computational performance. To stress the importance of proper cell size and clarify the trade-off between large and small cells, we should consider the following. Although a large cell size contains more points and could natively increase the localization accuracy of scanner positions by improving the feature expressiveness, it could also be more distorted by adjacent fuzzy point distributions, such as from overlapping scans or from projected points that belong to nearby interior objects. On the contrary, a small cell size may allow a finer and more reliable classification, but it would also result in an increased grid map resolution, leading to higher memory consumption and computational costs. Moreover, we should also take into consideration the fact that indoor environments typically contain many non-structural objects (such as furniture), which tend to clutter the environment and make the reliable detection of voting cells very challenging.

Under these conditions and depending on the application, the requirement of high resolution grid maps becomes a necessity. Therefore, in our pipeline the size of the grid elements in the generated 2D grid cell structure is defined according to the point sampling distance and the localization accuracy requirements of the application. As a default value, we used in our implementation a conservatively small cell size of $0.01m$, which gives a very reasonable trade-off between computational cost and high grid map resolution.

Then, to project the raw 3D points into the corresponding cells, we can simply calculate the orthogonal projection of points onto the 2D plane, given the orientation of the floor. In the simplified case, where the floor plane is aligned with a particular coordinate system plane, such as e.g. the x, y-plane, we can just drop a coordinate (e.g. the z).

Despite the optimal selection of cell's size, processing the grid structure in large-scale indoor spaces will be inevitably very challenging and time-consuming,

since the computation of cell features, as described in Section 5.3.2, requires frequent spatial range searches between the points contained in each cell. Considering also the demanding processing and data storage requirements for handling datasets of that size, we employed an efficient Kd-tree data structure [Elseberg et al., 2012], which is fully optimized for accelerating range and nearest-neighbor searches, and can optimally organize and classify the data points in the grid map.

5.3.2 Feature Extraction

After the creation of the grid map and the projection of points onto it, in the next step of our pipeline we need to classify the grid cells in order to identify the reliable voting cells. To do so, we extract a set of features from each cell, which is capable to discriminatively represent its spatial point distribution.

Emphasis has been given to the point distribution variations among the cells that lie either within the regions where the potential voting cells are (voting areas), or across the other regions of the grid map. The discrimination criteria between these regions rely mainly on the scanning characteristics mentioned in Section 2.2.1, such as the point distribution density in relation to the distance from the laser scanner and the angular scanning resolution (see also Figure 2.1). Additional variabilities caused by the projected points of adjacent indoor objects and overlapping scans are also factors that could easily cause various distortions to the point distributions in the 2D plane and could affect the final segmentation of cells. Thus, the correct choice of these features is very important.

In our method, we have adopted a computationally efficient statistical modeling approach that is broadly applicable to all cells of the grid map. Exploiting the information derived from applying spatial statistics and point distribution analysis on the grid cells, we create a set of features that most informatively characterize the spatial patterns of cells. Their selection was based on their capability to discriminate efficiently the most reliable voting cells from the other remaining cells of the grid, while other factors, such as their scale invariance, were also evaluated, targeting on features that are general enough to account for cell diversity and point distribution variability. For this reason, features relying mostly on point intensities, color, texture, shape or convexity tend to underperform under these conditions, since they do not manage to efficiently depict the differences between the two types of cells we are aiming for. On the other hand, descriptive features measuring the variance or the statistical dispersion of points inside a cell proved to be more suitable to characterize adequately a point distribution in the present context.

Thus, studying the spatial arrangements of points inside each cell, we have incorporated several features, whose combination permits efficient cell classification under the aforementioned general scanning conditions. The selected features

are:

Median nn-distance (ψ_1) This is the median length of all pairwise Euclidean distances between a point and its nearest neighbors in the cell. More precisely, denoting the spatial point distribution of each cell of size N by $\boldsymbol{\mathcal{P}}^s = (p_i^s : i \in \{1, 2, ..., N\})$ [Diggle, 2003], for any point $p_i^s \in \boldsymbol{\mathcal{P}}^s$ in cell s, the nearest neighbor distance (nn-distance) from p_i^s to all other points in $\boldsymbol{\mathcal{P}}^s$ is given by

$$d_i^s = d_i(\boldsymbol{\mathcal{P}}^s) = \min\big\{d(p_i^s, p_j^s) : p_j^s \in \boldsymbol{\mathcal{P}}^s, j \neq i\big\}, \tag{5.1}$$

where $d(.,.)$ is the Euclidean distance. Consequently, the median nn-distance is given by the following equation

$$d_{\text{med}}^s = \text{median}(\boldsymbol{\mathcal{D}}^s) \tag{5.2}$$

where $\boldsymbol{\mathcal{D}}^s = \{d_i^s\}$ is the set of all nn-distances between any point p_i^s and its nearest neighbor in cell s.

Mean nn-distance (ψ_2) This feature indicates the mean distance of all pairwise Euclidean distances between a point and its nearest neighbors in the cell. In a similar manner as the median nn-distance, it is given by

$$\widetilde{d}^s = \frac{1}{N}\sum_{i=1}^{N} d_i^s, \tag{5.3}$$

where the notation and constants are the same as above.

Height range (ψ_3) It represents the height difference between the two highest and lowest points in a cell, before their projection onto the 2D plane

$$d_{z-\text{range}}^s = \max_z(\boldsymbol{\mathcal{P}}^s) - \min_z(\boldsymbol{\mathcal{P}}^s), \tag{5.4}$$

where $\max_z$ and $\min_z$ indicate the maximum and minimum values along the z-axis of all points in $\boldsymbol{\mathcal{P}}^s$.

Standardized height dispersion (ψ_4) It quantifies the statistical dispersion of points along the z-direction inside the cell and penalizes the cells that exhibit large height variations, combining the performance of measures from descriptive and robust statistics. Specifically, it is expressed proportionally to the amount of points in each cell as

$$d_{\text{disp}}^s = \frac{d_{z-\text{range}}^s - \text{MAD}(\boldsymbol{\mathcal{D}}^s)}{N \cdot \text{STD}(\boldsymbol{\mathcal{D}}^s)}, \tag{5.5}$$

where STD is the standard deviation of all distances in $\boldsymbol{\mathcal{D}}^s$, and MAD is a robust estimator of the variability, indicating the median absolute deviation from the median [Huber and Ronchetti, 2009], which is computed by

$$\mathrm{MAD}(\boldsymbol{\mathcal{D}}^s) = \mathrm{median}\left\{\left|d_i^s - d_{\mathrm{med}}^s\right|\right\}. \tag{5.6}$$

Skewness of nn-distances (ψ_5) It measures the asymmetry in the distribution of nn-distances inside a cell and quantifies it as a value of spread. Typically the voting cells are skewed left, so using the Pearson's moment coefficient of skewness [An and Ahmed, 2008] we compute it as

$$\mathrm{skew}(\boldsymbol{\mathcal{D}}^s) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(d_i^s - \widetilde{d^s}\right)^3}{\mathrm{STD}(\boldsymbol{\mathcal{D}}^s)^3} \tag{5.7}$$

Kurtosis of nn-distances (ψ_6) It measures the normality of the distribution of nn-distances inside a cell and indicates how prone to outliers the cell is, by measuring the tail heaviness relative to that of the normal distribution. It is given by

$$\mathrm{kurt}(\boldsymbol{\mathcal{D}}^s) = \frac{\frac{1}{N}\sum_{i=1}^{N}\left(d_i^s - \widetilde{d^s}\right)^3}{\left(\frac{1}{N}\sum_{i=1}^{N}\left(d_i^s - \widetilde{d^s}\right)^2\right)^2} - 3 \tag{5.8}$$

where we subtract 3 from the raw kurtosis index to provide a figure relative to the normal distribution [An and Ahmed, 2008].

Histogram bin counts in XY (ψ_7) To compute these features, we first uniformly quantize the cell space into a particular number of bins based on d_{med}^s and we then generate a histogram for its x, y main coordinate axis. The number of bins along each axis direction that contain at least one point is used as a feature in our cell classification process, since it reveals the underlying shape of the point distribution.

Histogram bin counts in Z (ψ_8) Similar to the computation of the previous feature, we uniformly quantize the space between the minimum and the maximum height values of points contained in a cell based on d_{med}^s, and we then generate the corresponding histogram. The number of bins containing at least one point is used as the last feature in our classification method.

5.3.3 Outlier Removal

Before proceeding to the actual description of our cell classification approach, it is worth mentioning that, since we are performing a confirmatory data analysis

for quantifying the extent to which the point distributions of voting cells deviate from those of the other cells (a process similar to statistical hypothesis testing), the detection and omission of anomalous data points in point patterns is quite beneficial [Lu et al., 2003]. Although their amount in voting cells is not high, considering also the inherent ambiguities introduced to cell classification by the scanning process as described earlier, a filtering process for removing all points that spatially deviate a lot from the other points in a cell would be advantageous, increasing at the same time the discriminance of cell classification.

Therefore, we added to our pipeline a data cleansing stage, which detects and rejects *spatial outliers* that present local spatial instabilities with respect to the neighboring points. These outliers could bias and mislead the training process of our machine learning method, resulting in a less accurate model and ultimately in poorer results. For that reason, we exploit the pairwise nn-distances (Eq. 5.1) to reject all points in a cell which are greater than the γ-scaled MAD away from the median nn-distance (Eq. 5.2). The distances of inliers in that case are given by

$$\boldsymbol{\mathcal{D}}^s_{\text{filt}} = \{d^s_i \quad | \, d^s_i \leq \gamma \cdot d^s_{\text{outlier}}\} \quad \text{with} \quad d^s_{\text{outlier}} = \beta \cdot \mathrm{MAD}(\boldsymbol{\mathcal{D}}^s), \tag{5.9}$$

where the correction factor β is required to make MAD an unbiased estimate of the standard deviation for Gaussian data, and is equal to the reciprocal of the quantile function Φ^{-1} [Rousseeuw and Croux, 1993; Huber and Ronchetti, 2009; Pearson, 2001]. γ is the outlier rejection threshold and was set to 3 according to [Pearson, 2002; Pearson, 2001]. Figure 5.2 presents two examples taken from real-world datasets, where the effectiveness of this cleansing process in the point distributions of voting cells is demonstrated.

5.3.4 Cell Classification

The features presented in previous section (see Section 5.3.2) create a descriptive 8-dimensional feature vector $\Psi = (\psi_1, \psi_2, ..., \psi_8)$, which constitutes the point distribution signature for every given input cell. Next, we learn an SVM classifier to classify these signatures, expecting that spatially similar point patterns have similar signatures. The objective here is to compute a hyperplane which separates the two cell classes in feature space and classifies the new input data by deciding on which side of the hyperplane the new cell resides. Using a more formal notation, this task can be formulated as a binary supervised classification problem, where a given set of training samples (i.e. cells) $S = \{(\Psi_1, h_1), (\Psi_2, h_2), ..., (\Psi_s, h_s)\} \subseteq \Psi \times \mathcal{L}$, with $h_s \in \mathcal{L}=\{-1, +1\}$ being the target label set, can be used by a classifier for learning the distribution patterns of voting cells from the training samples S and predict reliably the label h_s for any unknown Ψ_s. In our approach, we used linear classifiers to learn the cell detectors, so that given the feature vector Ψ_s of a detected cell and the classifier $\mathbf{w}$, to be able to decide through the equation

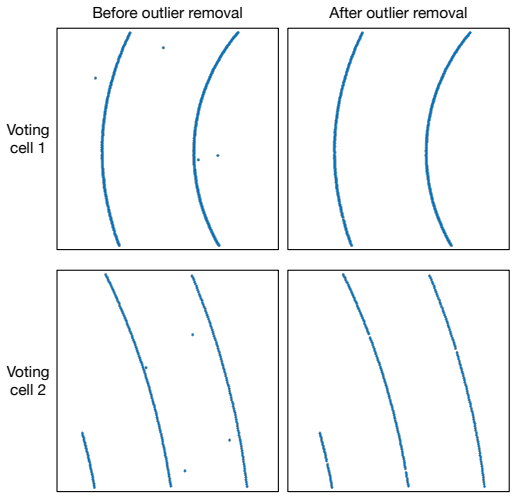


Figure 5.2: *Top view of two voting cells, showing the rejection of spatial outliers, i.e. noisy points causing local spatial instabilities.*

$\mathbf{w}^T \Psi_s > \tau$ to which semantic class the cell belongs to. The resulting labeled grid map, after the SVM classification, has a form similar to Figure 5.1(c).

We should notice here that, despite the effectiveness of the selected features for reliably classifying the voting cells, there might be some rare cases where some false positive voting cells will be detected. Since, however, these cells will be a minority in the grid map, their circle centers estimated by the next processing stage of our pipeline will deviate significantly from the majority of the good voting cells, and therefore, will cast outlying votes which will be rejected by the non-maximum suppression in the Hough voting space (see also Section 5.4.2).

In the next processing step, we will focus only on these classified areas, from which the scanner positions will be derived.

5.4 Scanner Position Reconstruction

After classifying the cells, all reliable voting areas in the grid map that conform to the scanning characteristics outlined in Section 2.2.1 have been detected. Focusing only on these areas and exploring further their point distributions, we retrieve more insight into their point patterns, from which in the end it will be possible to reconstruct reliably the scanner positions. Our strategy here is conceptually straightforward and is illustrated in Figure 5.3. Typically, the cells of voting areas contain points whose distributions form one or more curved-like shapes, since they constitute a small part of the characteristic circular point pattern around the

scanner position (see also Figure 2.1). Therefore, in this part of our pipeline we focus our analysis on this distinguishing attribute and we cluster the points of each voting cell based on their spatial point distance and point density (see Figure 5.3(a)). The resulted point clusters have a curved-like shape, to which we apply a circle model fitting algorithm as a circular scanning pattern hypothesis, in order to approximate the circular shape of the original scanning pattern (Figure 5.3(b)). For each scanning pattern hypothesis, we also compute the coordinates of its center, which constitutes a scanner position proposal, casting a weighted vote in a discretized Hough voting space from which the original scanner positions will be detected (Figure 5.3(c)). Repeating this model fitting process for all voting cells in the grid map, we generate for every cluster in each voting cell a circular point pattern hypothesis and compute the corresponding scanner position proposal. All these proposals cast votes to the Hough voting space, and after applying non-maximum suppression in it, we detect the cells where the scanner devices were located. In the final step of our pipeline, we reconstruct the exact scanner 2D coordinates (i.e. x and y) by applying mean-shift to the Gaussian KDE of the scanner position proposals in the detected cells. To reconstruct the height of laser scanners, although we could follow an equivalent analytical approach in 3D space in order to recover the exact z-coordinates, such a drastic increase in the computational cost might not be fully justified by its usefulness, since for the majority of tasks that rely on the scanner positions (e.g. wall detection, visibility testing, room space partitioning, etc.), the accurate localization of the scanner height is not absolutely necessary. Thus, we followed a more greedy approach and we computed the average height of scanner positions from the datasets we used for the training of the SVM classifier and we assigned it to the reconstructed 2D scanner positions as their third coordinate.

5.4.1 Clustering Voting Cells

In order to cluster the points inside the detected voting cells, we adopted a clustering method called *Gaussian density distance* (GDD) [Güngör and Özmen, 2017], which relies mainly on two features of data samples, the point densities and the pairwise point distances. This method is very efficient in clustering data points that their point distributions present variable densities or pairwise point distances, since it can efficiently differentiate between point clusters based on point densities and considering the spatial gradual variations in pairwise point distances. To compute the clusters, it uses first *Gaussian mixture models* (GMM) to find the centroids of the data space and then expands the cluster regions by evaluating various statistical measures, such as the variance, mean, deviation and Gaussian values of neighboring unclustered points. Also, the ability of this clustering method to create natural clusters of points, emulating the human perception without requir-

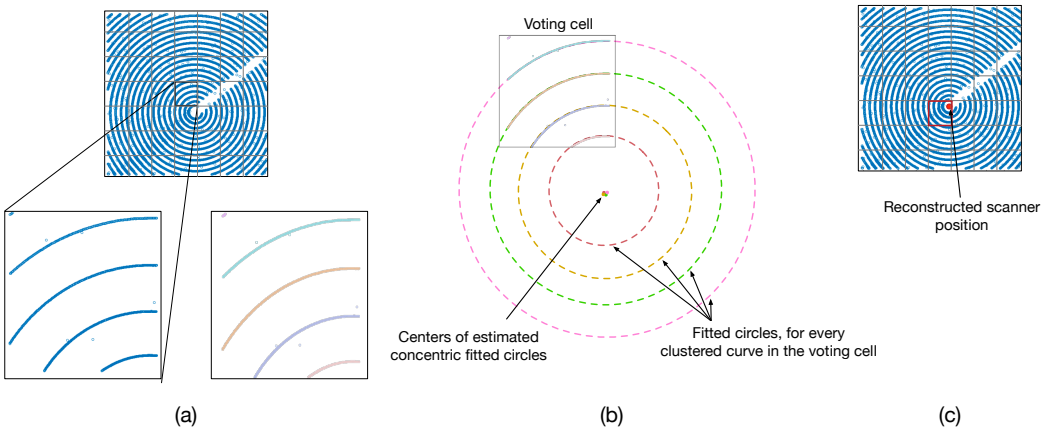


Figure 5.3: *(a) After the detection and classification of voting cell areas (here presented in blue), the points of voting cells are clustered using the GDD algorithm. (b) Then, applying a robust circle fitting method to each cluster, the corresponding circle centers are computed. (c) These centers cast votes for the scanner locations, allowing the detection of cells with the maximum proposals in the grid map (cell in red), from which the original scanner coordinates are reconstructed (red dot).*

ing any prior information, parametrization or supervision, constitutes its biggest advantage. An example of clustering a voting cell by applying the GDD method is presented in Figure 5.3(a).

After clustering the points $\boldsymbol{\mathcal{P}}^s$ of a voting cell s into l clusters $\mathcal{Q}^s = \{\mathcal{Q}_1^s, \mathcal{Q}_2^s, \ldots, \mathcal{Q}_l^s\}$, we fit a circle model to the points of each cluster in $\mathcal{Q}^s$ in order to approximate the circular scanning pattern from which they were generated. The problem of fitting a circle to a set of points can be approached using simple least squares (LS) [Pratt, 1987] by minimizing the sum of squared algebraic distances

$$\boldsymbol{E_a}(a, b, R_l) = \sum r_i^2, \tag{5.10}$$

where r_i is the distance between the coordinates (x_i, y_i) of a point $p_i^s \in \mathcal{Q}_l^s$ and the circle center (a, b) of the circle model fitted to the points of $\mathcal{Q}_l^s$, while R_l denotes the radius of this circle. However, in order to avoid the drawbacks introduced by the algebraic criterion [Gander et al., 1994], we preferred to use a geometric approach relying on an *orthogonal least squares* (OLS) fit and minimizing the sum of the squared geometric distances

$$\boldsymbol{E_g}(a, b, R) = \sum d(p_i^s, c_l)^2, \tag{5.11}$$

assuming that $\boldsymbol{E_g}$ is a continuous function. In the above equation, $d(p_i^s, c_m)$ stands for the Euclidean (geometric) distance between the point p_i^s in cluster $\mathcal{Q}_l^s$ and the

circle curve c_l generated by the points of $\mathcal{Q}_l^s$ (usually called residual). Using the difference-of-squares geometric error criterion [Calafiore, 2002], the problem in Eq. 5.11 has a closed form and the above distances can explicitly be defined by the formula

$$d(p_i^s, c_l)^2 = r_i - R_l \quad \forall R_l \geq 0 \quad \text{with} \quad r_i = \sqrt{(x_i - a)^2 + (y_i - b)^2}. \quad (5.12)$$

We should mention here that, although LS methods are sensitive to outliers and their results might be quite erroneous in cases with high noise, it was not necessary in our case to use more robust algorithms such as in [Yu et al., 2010; Nurunnabi et al., 2017] to accomplish the model fitting, since the point distributions of the detected voting cells contain only a very minor amount of noisy points due to our previous noise elimination processing step. A representative clustering example that exemplifies our argument is presented in Figure 5.3(a).

5.4.2 Scanner Position Selection

As was highlighted in the previous section, the conceptual basis of fitting a circle model to the clustered points of a voting cell is to allow each cluster to vote for a scanner location, considering that for their point pattern distributions we have already built high confidence through our strict classification process that they match the original circular scanning patterns of the measuring device.

Thus using a Hough voting scheme, our method casts for each fitted circle model a vote for the corresponding scanner position proposal into a discrete voting space. Typically, Hough voting establishes correspondences between object proposals (i.e. scanner locations) and a discrete confidence map supporting them, where the scores of all votes are accumulated and the regions with the highest voting scores in the map are detected using a suppression or mode-seeking method. In other words, this process is equivalent to finding the local maxima of intensities in an image, interpreting here as intensities the total number of proposals per grid cell. Figure 5.4 shows one such an example from a voting area of the building shown in Figure 5.1, where the cell intensities of scan position proposals are presented as a heatmap image.

Ideally, the above mentioned voting scheme produces good results, being capable to localize the correct scanner positions by detecting the voting cells in the grid map, which have the highest density of proposals. However, although this is the most prominent scenario for the detection of scanner locations, for generalization and completeness, there are also some more rare cases that we need to take into consideration. For example, the voting cells quite far away from the scanner locations tend to present point distributions that have very soft curved shapes, which do not allow for accurate circle fitting and hence, they do not cast consistent

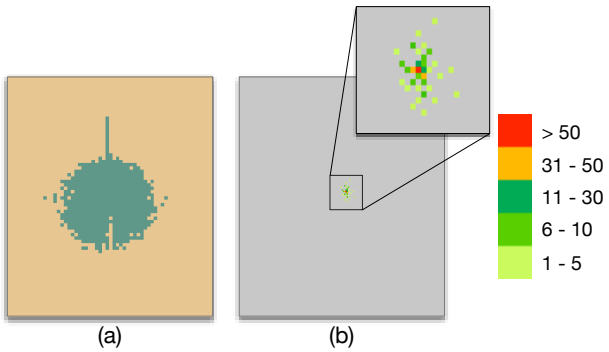


Figure 5.4: *(a) A classification example of voting cells (in green) around a scanner location for a region of the building model shown in Figure 5.1. (b) Heatmap of the estimated circle center locations from the voting cells, of which the majority of votes lies in the red cell.*

votes for the scanner positions. In addition, there are cases where the reconstruction environments are very challenging and cluttered, or the scanner locations are very close to the building's architectural elements, which introduce many distortions to point patterns and drastically reduce the number of detected voting cells to only a very few. In such cases, the scanner position proposals might compete with each other for the final scanner localization, fuzzifying the voting process. These and other similar cases are an exception to the general rule, but might cause inconsistencies or distortions in the voting process.

Thus we have enhanced the voting process by using a simplified weighted voting model, which penalizes misleading or inconsistent votes and rewards the most reliable ones. Specifically, it assigns larger weights to the votes originating from point clusters closer to the scanner proposals, as well as from those presenting a smaller magnitude of residuals from the circle fitting model. Hence, the weights in our voting scheme are computed by

$$w_l = \frac{k_l}{RMSE_l} \cdot e^{-R_l} \tag{5.13}$$

where k_l is a proper scaling factor and $RMSE$ is the root mean squared error, which quantifies how well the LS method fits the data. It also expresses the goodness-of-fit of the fitted circle model to the data points of the cluster, and is computed by

$$RMSE_l = \sqrt{\frac{\sum_{i=1}^{N_l} d(p_i^s, c_l)^2}{N_l}}, \tag{5.14}$$

where N_l is the total number of points in cluster $\mathcal{Q}_l^s$. We should also mention

here, that although the implemented voting scheme seems to be quite simple, it has been proven to work very robustly in practice, covering effectively all possible cases, and thus it was not necessary to follow a more complex probabilistic Hough voting scheme, such as the ones presented in [Leibe et al., 2004; Gall et al., 2011].

In the next step of our pipeline, we use non-maximum suppression to detect the regions in the grid map that correspond to the locations with the maximum responses in the confidence map of the Hough space, although any other mode-seeking method could be also used for this purpose. However, we preferred to use non-maximum suppression, since it has been proven to perform better than other mode-seeking approaches when analyzing the Hough space [Woodford et al., 2014].

The result of the above-mentioned process is a collection of cells in the grid map, in which the scanner positions are located. Based on the default cell size of 1cm we have set in our implementation (see also Section 5.3.1), the localization of scanner positions has already reached an accuracy of $\pm$1cm. However, we can further analyze the distribution of scanner position proposals inside these cells and increase the accuracy of our method by computing the exact scanner coordinates within the cell. This is achieved by using a Gaussian kernel density estimate (KDE) for analyzing the probability density function of the proposals inside these scanner location cells (such as the cell marked in red in Figure 5.3(c)) and then by applying mean-shift to find the mode of this density function. Finally, finding the nearest neighboring proposal to the computed mode value, we reconstruct the final scanner coordinates with an accuracy in the tenths of a millimeter.

5.5 Results

This section presents the experimental results of our novel pipeline and discusses the insights derived from them. In our test suite, we focused mainly on testing our approach on datasets used by the state-of-the-art methods in the field of 3D BIM reconstruction, in order to better show the generalization and applicability of our method. Therefore, our experimental setup is composed of real-world and synthetic datasets from building interiors, including several models as used in the related literature, such as from [Mura et al., 2014; Ikehata et al., 2015; Ambrus et al., 2017; Mura et al., 2016]. An overview of all datasets and their statistics is provided in Table 5.1. The real models from [Mura et al., 2014; Mura et al., 2016] were acquired by high-quality terrestrial laser range-scanner (LiDAR) without any manual intervention, while the acquisition characteristics of the other datasets are according to the descriptions provided in the corresponding works. With respect to the synthetic models that we used in our test bench, they were generated manually by [Mura et al., 2014] using a 3D modeling software and were virtually scanned

from several positions, while their depth measurements were artificially distorted by additive Gaussian noise (with $\sigma = 1mm$), in order to simulate realistically the quality of real scans.

In general, the buildings of the datasets we used for evaluating our method were carefully selected, in order to present diverse properties in architectural design and shape, while they include working and domestic environments with various interior objects. Specifically, they include indoor environments ranging from individual rooms (e.g. 'Office G82') to entire floors of buildings (e.g. 'Apartment 1'), which present different scanning characteristics, highly overlapping and extremely subsampled regions, as outlined in column 'Semantics' in Table 5.1.

Implementation The implementation of the software prototype of the proposed approach was based on Matlab and C++, using mainly C++ and MEX files for the computationally demanding tasks. All tests were performed on a Macbook Pro with an Intel Core i7 (2.5GHz), 16GB DDR3 RAM and an NVIDIA GeForce GT 750M. The number of scans (listed in column 'Scans' of Table 5.1) for the datasets we used ranges from 2 to 16 and provides a rough overview of the size of the area covered by each dataset.

With respect to the classification of cells, the training of our machine learning approach was based on a labeled set of cells, carefully selected for this task, while for their classification, we used linear classifiers trained via LIBLINEAR [Fan et al., 2008]. To build the cell models for each cell class, we used a representative collection of manually labeled samples, which were selected by a variety of datasets captured by laser scanners from different manufactures and from various indoor environments with slanted walls, arbitrary wall orientations and different levels of clutter and occlusions. Provided these training segmentations, our algorithm was able to automatically learn a discriminative model to segment a new cell, without any user intervention.

Quantitative and qualitative evaluation The datasets we used in our test bench were intentionally selected due to the differences and variabilities they present in their acquisition conditions and post-processing steps (e.g. registration, subsampling, etc.), according to the descriptions provided in [Mura et al., 2014; Ikehata et al., 2015; Ambrus et al., 2017; Mura et al., 2016]. Thus, each dataset presents unique characteristics that challenge our reconstruction pipeline and prove the general validity and performance of our method. Specifically, Figures 5.5, 5.6 and 5.7 show the reconstructed scanner positions from the datasets used in our test suite, while Table 5.2 shows the coordinates of the original and the reconstructed scanner positions along with their absolute and relative distance errors. We also present in Figure 5.8 the box plots of the corresponding datasets,

Dataset	#Points	#Scans	Type	Semantics	#Total C.	#Voting C.	Reference
Office G82	22.7M	2	LiDAR	R, HSD, HOR	0.153M	3668	Own
Cottage	12.4M	7	LiDAR	B	0.404M	3816	[Mura et al., 2016]
Apartment 1	7.2M	16	LiDAR	B, ES	0.662M	114	[Ikehata et al., 2015]
Building 8	19.5M	7	LiDAR	B, HSD, HOR	0.647M	83074	[Ambrus et al., 2017]
Penthouse	22.2M	8	LiDAR	B, HSD, HOR	0.732M	33407	[Mura et al., 2016]
Synth1	19.4M	7	Synth.	B, HOR	0.771M	53677	[Mura et al., 2014]
Synth2	19.3M	7	Synth.	B, HSD, HOR	1.82M	57649	[Mura et al., 2014]

Table 5.1: *Source point cloud information (no. of points, no. of scanner positions, type, semantics) and statistics (total number of grid and voting cells). Semantics describe some characteristics of the model, i.e. if it is a room (R) or building (B), and if it presents high scan density (HSD), high overlapping regions (HOR) or if it was extremely subsampled (ES).*

in order to graphically depict the total variability and the degree of dispersion (spread) of absolute error between the original and the reconstructed scanner positions, according to the following equations.

$$Err_a = \sqrt{(x_{orig} - x_{rec})^2 + (y_{orig} - y_{rec})^2} \tag{5.15}$$

$$Err_r = \frac{Err_a}{\sqrt{(x_{orig} + y_{orig})^2}} * 100\% \tag{5.16}$$

Figure 5.5(a) shows the 'Cottage' [Mura et al., 2016] building model composed of multiple rooms, slanted ceilings and vertical wall surfaces, as well as its corresponding grid map after the detection of the voting cells (in green). This environment challenges our method due to its steep sloped ceilings, because any potential circular point pattern depicted in the ceiling of the building would create an ellipsis when projected onto the 2D grid map, which consequently will lead to the distribution of votes over a line. However, this voting pattern is not affecting the accuracy of the reconstruction results, since the processing steps of our pipeline related to mode seeking and non-maximum suppression are not influenced by these votes. One example for such a characteristic region from this dataset is presented in Figure 5.4, where the votes for the scan positions are also presented as a heatmap for the corresponding voting area. The reconstructed scanner positions are presented in the bottom row of Figure 5.5(a) as red spheres in 3D space.

Another, even more challenging environment with many slanted walls, more structural details such as the window alcoves, and less regular room shapes is

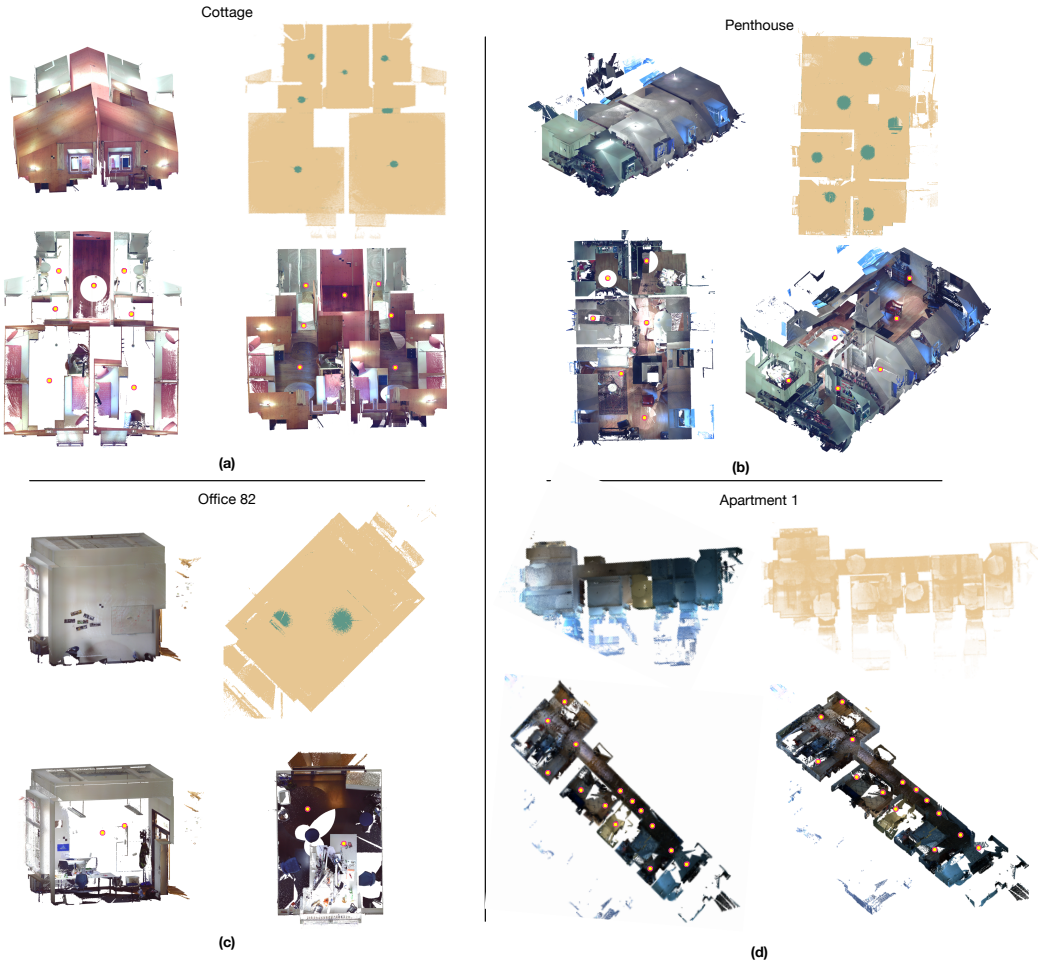


Figure 5.5: *Scanner position reconstruction results for Cottage, Penthouse, Office G82 and Apartment 1. For each dataset, we show the source point cloud (top left), the voting cells in the grid map in green (top right), and two views showing the reconstructed scanner positions with red spheres (bottom row for each dataset).*

presented in Figure 5.5(b). Despite the high levels of clutter and the distortion of point pattern from the slanted walls, our method manages to correctly reconstruct all scanner positions, as it is depicted in the bottom row of Figure 5.5(b). For the numerical accuracy of all models see also Table 5.2.

The third dataset we used for evaluating our method is the 'Office G82' presented in Figure 5.5(c). This office room represents an exceptionally difficult setting for laser scanning since it is quite small (approx. 4.80×4.10m) and includes many objects and furniture that not only provide very limited space for

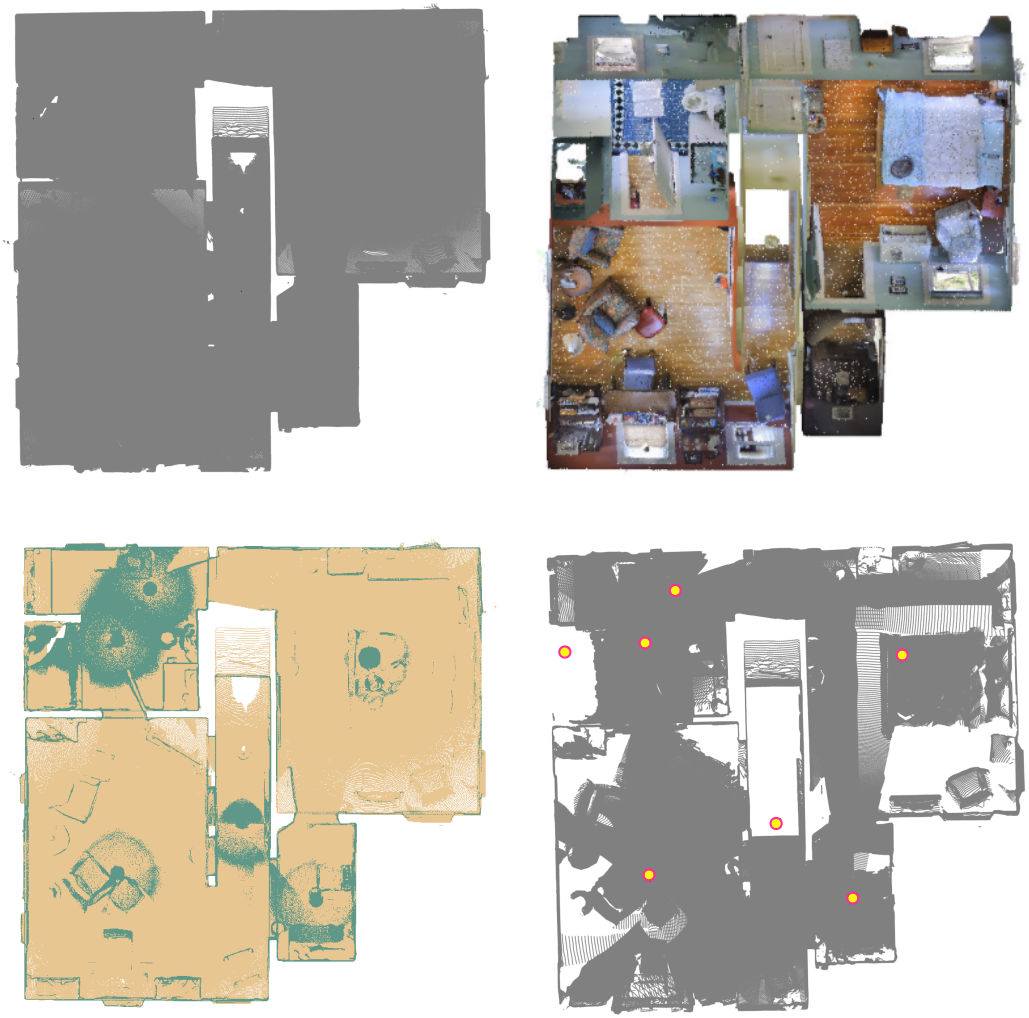


Figure 5.6: *Scanner position reconstruction results for the Building 8, showing the input dataset (figures in top row), the computed voting cells in the grid map (bottom left), and the reconstructed scanner positions marked with red spheres (bottom right).*

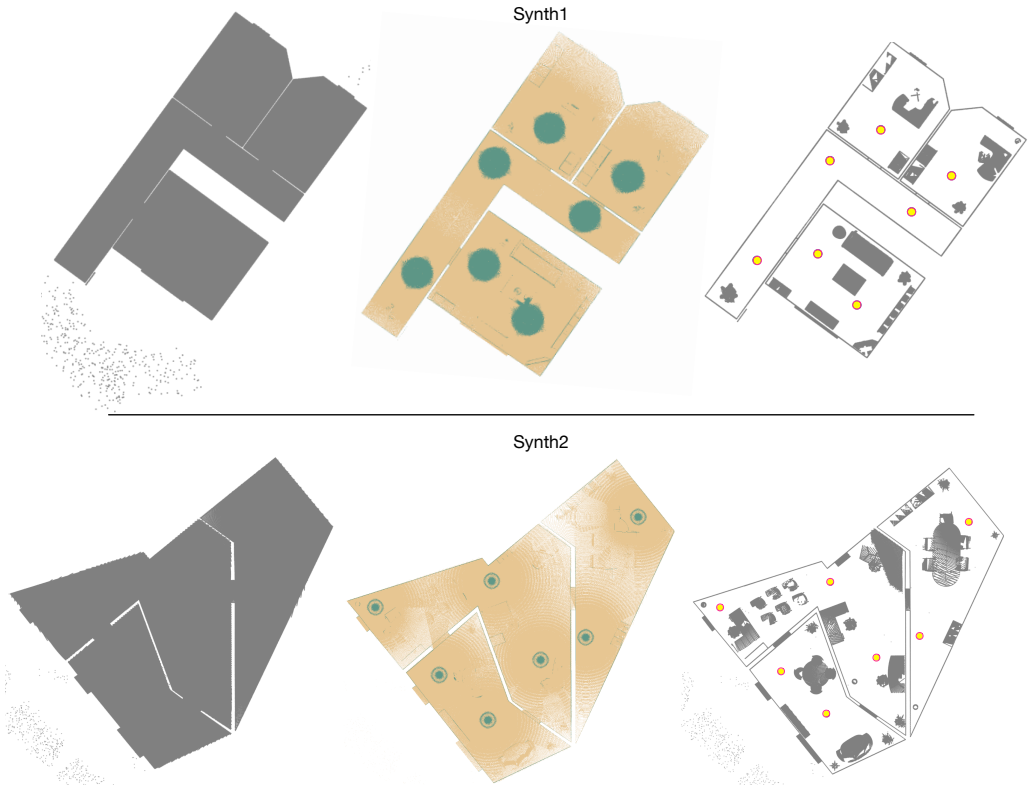


Figure 5.7: *Scanner position reconstruction results for the synthetic datasets Synth1 and Synth2. For each dataset, we show the source point cloud (left), the voting cells in the grid map (middle), and the reconstructed scanner positions marked with red spheres (right).*

scanning the room, but also cause a highly cluttered scene with many occluded regions. Typically, under these demanding conditions and considering the size of the room, one scan from a central position would be adequate to capture its shape and structure. However, in order to challenge and test our method under scanning conditions where scanners are placed very close to each other creating dense and highly overlapped regions, and where the environment has limited flat ground surfaces, we scanned the room twice, positioning the scanners in a horizontal distance of approximately $1.5m$ and a height distance of about $0.80m$ between them. As it is depicted in Table 5.2, the reconstructed scanner positions are accurately computed, despite the challenges and the difficult scanning conditions. Note that, the stripes evident in the circular shape formed by the voting cells in the top right figure of Figure 5.5(c) were caused mainly by the ceiling mounted light devices of the room, which act here as occluders that scatter the laser beam and aggravate

the scanning pattern. These lighting devices are evident in the bottom left figure of Figure 5.5(c).

We should notice also here that, in general, clutter and outliers outside of the building are effectively ignored by our method and they do not contribute to the voting process for the scanner positions. However, clutter inside the building can affect the local distribution of points when projected onto the 2D grid plane and might distort the point pattern. Therefore, it is quite possible that this typical and uniform circular shape formed by the voting cells around the potential scanner positions, which is clearly evident for example in Figures 5.5(a) and 5.5(b), might be distorted and corrupted, and only a few randomly scattered voting cells may be detected. However, even in this extreme case, these few voting cells could be sufficient for our pipeline to reconstruct accurately the original scanner positions, since our method theoretically requires only one reliable voting cell per scanner positions in order to recover it.

The datasets in Figures 5.5(d) and 5.6 constitute typical examples of such cases. Especially the 'Apartment 1' dataset in Figure 5.5(d) obtained by Ikehata and colleagues [Ikehata et al., 2015] constitutes the most challenging dataset in our test bench because the point cloud was originally discretized with a voxel size of $0.012m$, which is almost equal to our grid cell size. This discretization restricts the cell classification process and only very few voting cells can be identified (i.e. 0.017% of the total number of grid cells). Nonetheless, the projection of all points in 3D space onto the 2D grid plane created sufficient point patterns necessary for the detection of some voting cells and consequently for the localization of the scanner positions, although two of the original scanner positions were not possible to be reconstructed due to the limited number of voting cells.

Figure 5.6 presents a dataset from [Ambrus et al., 2017], where a building representing the interior of a house was scanned from 7 scan positions with very high resolution. Because of its high scanning resolution, many voting cells were detected (more than 10^4) in different regions of the grid map, but due to the existence of occlusions and clutter near the LiDAR scanners, the detected voting areas do not all form the typical circular shape around the scanner positions. Nevertheless, our method can estimate the actual scanner positions very accurately, based on the detected voting cells, as it is verified in Table 5.2.

In our testing setup we additionally used two synthetic datasets, which represent different room settings and are shown in Figure 5.7. These datasets present more general building shapes and were virtually scanned from several positions to simulate the results of 3D laser range scanning, as it is described in [Mura et al., 2014]. Although these datasets were artificially corrupted with additive Gaussian noise, our method managed to correctly recognize the adequate number of voting cells in order to accurately reconstruct the scanner positions.

In Figure 5.8 we show the boxplots of error variability for each dataset used

in our test bench. The dispersion of errors shows that the error spread was less in 'Building 8', mainly due to the high number of voting cells detected, which increased the convergence to the original scanner positions. For the 'Office G82', no box plot can be created due to the low number of scan positions (i.e. only two), while for the very challenging dataset 'Apartment 1', its absolute errors are in average one order of magnitude higher than the errors of the other datasets and therefore its boxplot falls outside the depicted range in Figure 5.8. Nonetheless, the errors of the estimation of scanner positions still remain very low as shown in Table 5.2.

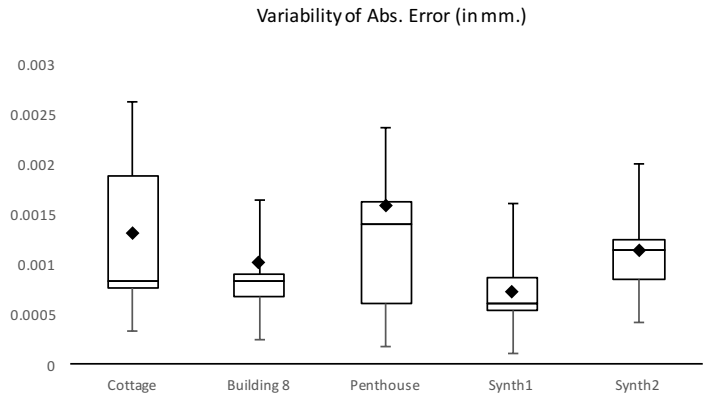


Figure 5.8: *Box plots showing the error variability (in mm) of the reconstructed scanner positions with respect to the true positions for the datasets of our test bench.*

5.6 Conclusion and Discussion

In this chapter, we have presented the third contribution of this thesis for re-engineering and reconstructing the original scanner viewpoint positions from raw and merged point clouds. To the best of our knowledge, this is the first method in literature that can lift the assumption introduced by the majority of 3D reconstruction methods until now, that the true scanner positions should be known a priori, in order the generation of the 3D models of the scene to be performed successfully. Our method relies on the intrinsic scanning characteristics of the point cloud acquisition process, and following an efficient and robust classification and segmentation process, it is capable to identify reliable areas which allow for the accurate detection and reconstruction of the original scanner positions.

Moreover, our evaluation process has proven that our approach allows the accurate localization of LiDAR devices under very challenging indoor environments and without requiring any prior knowledge about the environment or the dataset. Also, the proposed reconstruction method is independent from the laser scanner

manufacturers and can efficiently be applied to the majority of real-world and large-scale indoor point clouds, where the original scan positions are typically lost and not available.

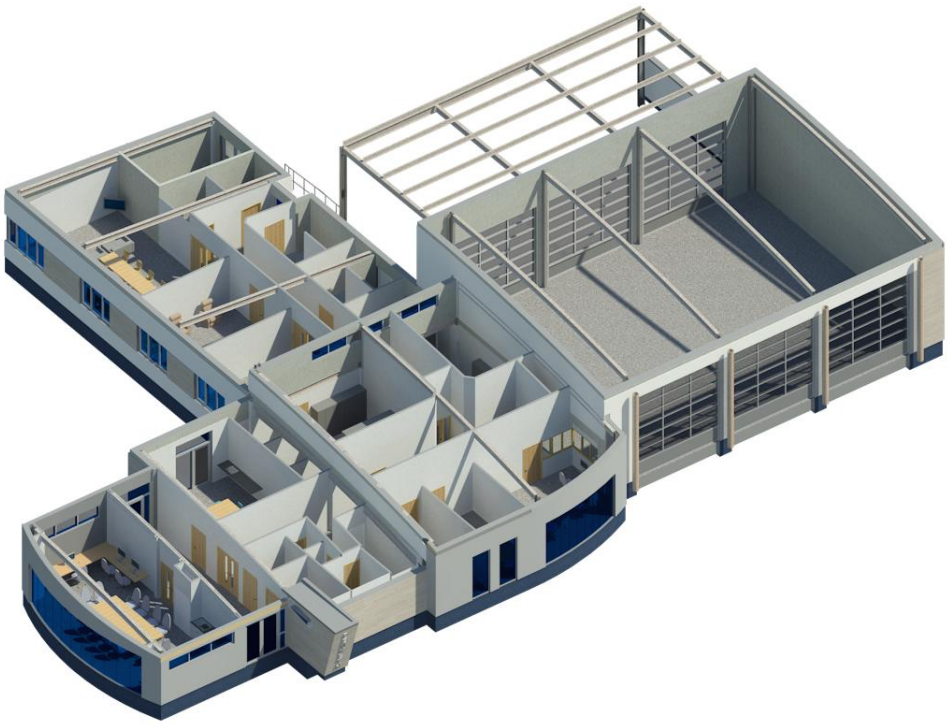
Despite, however, its robustness and efficiency, our method might present some limitations which could restrict its applicability, while it provides also some space for further improvements. For example, the classification stage of 2D grid cells in our approach relies on a machine learning method, whose classifiers were trained by a selected training set of labeled cells. Although these cells were obtained from various datasets which present different scanning patterns, point densities and variable point distributions, this training set may not be adequate enough to cover all possible real-world cases. Therefore, in a future work, more cell samples from new diverse datasets could be added to further enrich the training dataset, while a cross validation step could be also added to our pipeline for maximizing further the learning set size. Another potential limitation is introduced by the selected features used for the classification process. Despite their expressiveness and efficiency to classify a wide range of cells with various point distribution patterns, there are cases where false positive voting cells are generated. To eliminate the false positive rate, we could introduce more complex and expressive features in the feature vector, with the expense however of an equivalent increase to the computational cost during the training and classification stages. Finally, the fixed cell size, which determines the resolution of the grid map, defines also the exposure of the spatial point pattern to the feature evaluation process. Although it can be set by the user according to the application requirements, it limits our method's flexibility. Therefore, a method capable to adapt dynamically the cell size based on the available local point information would be beneficial for our pipeline and could increase drastically its adaptability to the scanning characteristics of the input datasets.

Dataset	X (orig.)	Y (orig.)	X (rec.)	Y (rec.)	$Abs.Err$	$Rel.Err(\%)$
Office G82	2.99998	2.9979	2.99925	2.9981	0.000757	0.0178
	1.48297	2.98107	1.48286	2.98112	0.000121	0.0036
Cottage	7.72902	4.65971	7.72914	4.66047	0.000769	0.0085
[Mura et al., 2016]	3.70268	5.30020	3.70299	5.30033	0.000336	0.0052
	5.86313	5.66134	5.86367	5.66189	0.000771	0.0095
	7.92553	6.10199	7.92515	6.10037	0.001661	0.0166
	4.34241	2.09008	4.34194	2.09266	0.002623	0.0544
	7.04545	2.93727	7.04589	2.93798	0.000833	0.0109
	8.62153	3.70510	8.62015	3.70668	0.002098	0.0224
Apartment 1	16.499997	9.451058	16.44170	9.66855	0.225168	1.1842
[Ikehata et al., 2015]	15.496077	11.086808	15.59300	11.16430	0.124093	0.6513
	11.857877	12.011518	12.36480	12.53210	0.726620	4.3050
	14.809787	8.292258	14.89220	8.39030	0.128079	0.7546
	13.292587	9.366978	13.36700	9.31910	0.088485	0.5441
	11.583787	8.118498	11.41410	8.32320	0.265888	1.8797
	9.067897	6.847718	9.09904	6.89560	0.057119	0.5027
	8.390557	8.239218	8.4232	8.23150	0.033543	0.2852
	10.011457	9.775588	10.01600	9.85050	0.075050	0.5364
	6.918127	7.732528	6.92950	7.72610	0.013064	0.1259
	7.309647	5.616968	7.31920	5.66040	0.044470	0.4824
	3.901797	4.017698	3.95520	4.07570	0.078842	1.4078
	2.631197	3.006268	2.65060	3.03410	0.033928	0.8492
	3.038797	5.914948	3.06470	5.91950	0.026300	0.3955
	8.159137	6.128218	-	-	-	-
	5.987997	5.022608	-	-	-	-
Building 8	6.06398	5.0333	6.06468	5.0339	0.000922	0.0117
[Ambrus et al., 2017]	1.81398	8.6333	1.81368	8.6339	0.000671	0.0076
	3.91397	7.7833	3.91417	7.7825	0.000825	0.0095
	2.36397	3.8833	2.36437	3.8862	0.002927	0.0644
	5.16398	9.0833	5.16342	9.0837	0.000688	0.0066
	1.81398	4.7333	1.81391	4.7335	0.000212	0.0042
	0.51398	4.8333	0.51311	4.83321	0.000875	0.0180
Penthouse	4.499927	1.799518	4.498257	1.799142	0.001712	0.0353
[Mura et al., 2016]	3.224057	4.352828	3.224586	4.354123	0.001399	0.0258
	4.643759	7.288818	4.643164	7.28859	0.000637	0.0074
	1.661347	7.556028	1.662534	7.551153	0.005017	0.0649
	4.588838	10.907068	4.588724	10.90727	0.000232	0.0020
	2.431447	9.897168	2.432537	9.896112	0.001518	0.0149
	6.142037	5.685188	6.141467	5.685274	0.000576	0.0069
Synth1	2.90416	4.4687	2.9052	4.4681	0.001201	0.0225
[Mura et al., 2014]	0.90416	2.6687	0.90426	2.6691	0.000412	0.0146
	0.90416	8.0687	0.90443	8.0683	0.000483	0.0059
	5.10416	8.3687	5.10454	8.36823	0.000604	0.0062
	5.60416	10.6687	5.60412	10.6681	0.000601	0.0050
	1.90416	10.4687	1.90427	10.4676	0.001105	0.0104
	5.60416	3.6687	5.60397	3.6693	0.000629	0.0094
Synth2	10.477	5.9645	10.4764	5.9641	0.000721	0.0060
[Mura et al., 2014]	7.227	8.9645	7.2281	8.9639	0.001253	0.0109
	2.977	13.4645	2.9788	13.4651	0.001897	0.0138
	10.727	15.2145	10.7281	15.2148	0.001140	0.0061
	13.977	9.9645	13.9779	9.9641	0.000985	0.0057
	16.977	11.4645	16.9767	11.4651	0.000671	0.0033
	20.477	19.4645	20.4764	19.4634	0.001253	0.0044

Table 5.2: *Original and reconstructed scanner position coordinates for Cottage, Office G82, Apartment 1, Penthouse, Building 8, Synth1 and Synth2, along with their absolute and relative errors.*

C H A P T E R 6

CONCLUSIONS



6.1 Summary

Recent years have witnessed an increased demand for the reconstruction and modeling of building interiors, mainly due to their potential applications in many diverse domains, where new horizons have been opened for the more efficient and productive exploration and organization of indoor environments.

In this thesis, we addressed some of the most important challenges still present in the state-of-the-art methods, proposing improvements and further enhancements to various parts of the reconstruction and 3D modeling process of building interiors. In particular, based on the current research methods, our work considers both stages of the traditional 3D reconstruction pipeline (i.e. the acquisition and 3D modeling stages) and sets specific research goals, contributing to both stages. Specifically, our work focuses on: improving the efficiency of 3D data acquisition, trying to bridge the gap between the highly accurate but slow laser scanners and the faster but less accurate stereo vision systems; detecting, reconstructing and semantically annotating the architectural wall elements of building interiors; reconstructing automatically the original scanner viewpoint positions from raw point clouds. These specific research objectives are achieved in this thesis with three research contributions, which are presented in Chapters 3, 4 and 5. Each of them implements a processing pipeline which contributes to specific parts of the reconstruction process in accordance with the objectives set and as illustrated in Figure 1.2.

As a first research contribution, this thesis focuses on the acquisition stage of the reconstruction pipeline and proposes a novel hardware-based stereo reconstruction method, which is aligned with the requirements of real-world applications and is capable to reconstruct the 3D information of the scene with adequate accuracy under strict time constraints. Our method considers two color stereo images as the initial input representation for the scene to be reconstructed and applies to them a simple filtering process, in order to smoothen the pixel intensities and reduce the introduced noise from the sensor devices. Then, a correlation-based matching cost algorithm was used for measuring the similarity of points in the filtered stereo images, producing a 3D cost volume called disparity space image (DSI). Considering the many unpredictable factors that can affect the stereo matching process and subsequently the quality of the extracted depth images, we relied on Discrete Dynamical Systems and we developed an efficient cellular automata (CA) algorithm, which eliminates the unjustified cost variations in DSI and refines the computed matching cost values. In the last stage of our approach, a similarity accumulator was applied to find in the refined DSI the corresponding points and indicate the most likely depth value for each pixel in the image plane. To achieve high performance and fast computations, we presented also in the same contribution an efficient parallel-pipelined hardware architecture that implements

the proposed stereo method on a custom FPGA device, allowing the extraction of high resolution range images in short processing times. The resulting depth maps from our system are suitable for a wide range of indoor applications, especially in cases where it is more important to reconstruct adequately and fast the scene structure, rather than to have a highly accurate but slow reconstruction of it. For these applications, the proposed stereo reconstruction method is ideal, capable to produce high definition range images in real-time speeds.

In the second contribution of this thesis, we focused on the detection and reconstruction of the main architectural wall elements of indoor environments, mainly due to their importance for the faithful recovery of building's *as-is* state. Thus, we introduced in Chapter 4 a novel pipeline, which is capable to reconstruct automatically the architectural wall elements of building interiors, such as the windows and doors, under very challenging conditions and significant amounts of clutter and occlusion. Assuming that the architectural 3D model of the indoor environment has been already reconstructed, our approach extracts the planar surfaces that correspond to the structural building elements (e.g. ceiling, floor and walls) and for each wall surface it approximates its outline shape using an $alpha$-shapes algorithm. In order to get the regularized boundaries of the wall surfaces, a robust multi-line model fitting algorithm is applied to the outline polygons, and the generated line model space is then clustered by a mean-shift clustering technique. From each cluster, the best representative mode line is later selected and a 2D cell complex structure is formed on the wall surface from the representative lines of all clusters. After enhancing the wall surface by recovering its occluded regions by an occlusion detection mechanism, we represent the 2D cells of the cell complex by means of an adjacency graph, in which a graph-cut based optimization method is applied to partition the wall surface and semantically segment the wall openings. Due to its characteristics and the inability of the majority of current modeling pipelines to reconstruct the wall openings, our approach could be beneficially embedded in any of these pipelines, providing enhanced modeling capabilities and enriching semantically their results. Additionally, being independent from any manual tuning and without relying on imagery or depth data, our approach can operate fully automatic and be more generic than the other related methods.

Different from the other two, our last research contribution attempts to address the problem of retrieving the real scanner viewpoint positions from raw point clouds. This is a vital information needed to almost all modeling methods in literature and until now no method was capable to recover it from the raw datasets. Thus, it was always assumed to be known a priori, introducing a hard restriction to the modeling process and requiring human intervention for its reconstruction. Our work, which is described in Chapter 5, allows the automatic computation of the original scanner position coordinates, exploiting only the information derived

from the raw point data. Based on the intrinsic scanning characteristics of LiDAR devices, our method relies on a descriptive feature vector, in order to classify the areas in the point cloud and identify the reliable regions which will allow for the detection and accurate reconstruction of the scanner positions. Due to its robustness and independence from laser scanner manufactures, our approach can be applied to almost all real-world LiDAR point clouds and in any indoor environment, while being able to reconstruct automatically the original scanner positions, it would be very beneficial to the majority of current modeling pipelines that target on reconstructing the architectural structure of indoor environments, on detecting occlusions or on reconstructing the architectural wall elements. Additionally, thanks to its unique capabilities, this work advances the state-of-the-art in the field and brings current modeling pipelines a step forward towards their fully automatic execution for solving the scan-to-BIM problem.

6.2 Directions for Future Work

The contributions of this thesis address several fundamental issues evident in the reconstruction and modeling process of building interiors. Nevertheless, our work revealed some new problems that have not been addressed yet, while during our engagement with this topic we realized several future research directions and ideas, which are discussed below.

- **Enhanced matching cost algorithm for stereo reconstruction**

  In Chapter 3 we presented a stereo reconstruction method, which relies on a local-based window matching algorithm (SAD) for computing the correlation costs. Although the proposed stereo reconstruction method provides robustness and adaptivity, which is confirmed also by the extracted results presented in Section 3.6, its local-based nature creates some errors in challenging and low-textured areas in the scene. It would be therefore beneficial to use another matching cost algorithm for generating the DSI, such as the semiglobal matching (SGM) method [Hirschmuller, 2008], which in combination with our CA-based optimization process, could produce more accurate and plausible range maps.

- **Improving the effectiveness of CA rules**

  In our first contribution, the DSI refinement process relies on a CA structure, which applies 3 different transition rules to the 3D space of DSI. As already mentioned, the main idea behind these CA rules is that each rule should be applied to specific regions in DSI, in order to find and replace, with the most reliable way, the unreliable matching cost values. Although the performance

and the effectiveness of each rule is quite satisfactory, it is inevitable that some wrong replacements will occur during this refinement process. Thus, in a future work, we could modify the CA rules, in order each rule to operate as a complemented rule for the previous one, eliminating the wrong replacements and improving further their effectiveness.

- **Refinement of wall plane segmentation**

  In the second contribution of this thesis, we proposed a method to segment the wall surfaces of building interiors using a graph-cut optimization technique applied to the cell complex of the wall plane. Despite its efficiency, in very challenging environments and under specific conditions, the segmented wall surface might include some misclassified regions, as discussed in Section 4.5. Therefore, an approach capable to enhance the segmentation results by eliminating these inconsistencies would be desirable. A possible enhancement would be to interpret the cell complex as a grid of cells and apply on it a refinement algorithm based on cellular automata. Following such an approach, the segmentation results from the graph-cut method would be re-evaluated and any misclassified cells could be re-labeled based on the applied CA transition rules.

- **Reconstruction of architectural wall elements with curved frames**

  Our method for reconstructing the architectural wall elements of building interiors relies on the segmentation of a 2D cell complex structure, which is mainly composed by quadrilateral cells. Using such a structure, wall openings with curved or arched frames cannot be faithfully represented and thus will be only partially reconstructed. Therefore, it would be useful to study an approach that could model these shapes more consistently before the creation of the cell complex, using for instance RANSAC-based methods [Schnabel et al., 2007] to fit models of curved and not only linear primitives to the wall opening frames.

- **Dynamically adapting the grid cell size for reconstructing the scanner positions**

  Despite its novelty and generalized applicability, our pipeline for reconstructing the original scanner viewpoint positions presents also some space for further improvements. As was mentioned in Section 5.3.1, the fixed cell size introduces some restrictions, and although it can be optimally set by the user according to the localization accuracy requirements of the application, it limits our method's flexibility and adaptability, while it affects also the computational performance. Therefore, in a future work, the fixed grid map could be replaced by a quadtree or some other spatial data structure, adapting dynamically the cell size of the 2D grid map based on the available local point information.

BIBLIOGRAPHY

[adi, 2015] (2015). Adisk bim model (https://www.asidek.es).

[CGA, 2018] (2018). The Computational Geometry Algorithms Library (CGAL) (https://www.cgal.org).

[pmd, 2018] (2018). PMD Tec (https://www.pmdtec.com).

[sce, 2018] (2018). SceneScan (https://nerian.com/products/scenescan-stereo-vision).

[VTK, 2018] (2018). The visualization toolkit (VTK) (https://www.vtk.org).

[zed, 2018] (2018). ZED stereo camera (https://www.stereolabs.com).

[Adan and Huber, 2011] Adan, A. and Huber, D. (2011). 3D reconstruction of interior wall surfaces under occlusion and clutter. In *In Proceedings Symposium on 3D Data Processing, Visualization, and Transmission*, pages 275–281.

[Adan et al., 2015] Adan, A., Quintana, B., Vazquez, A. S., Olivares, A., Parra, E., and Prieto, S. (2015). Towards the automatic scanning of indoors with robots. *Sensors*, 15:11551–11574.

[Adan et al., 2013] Adan, A., Xiong, X., Akinci, B., and Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31:325–337.

[Agarwal et al., 2011] Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., and Szeliski, R. (2011). Building Rome in a day. *Communications of the ACM*, 54(10):105–112.

[Akca, 2003] Akca, D. (2003). Full automatic registration of laser scanner point clouds. *Measurement Techniques VI*, 1:330–337.

[Ambrosch et al., 2009] Ambrosch, K., Humenberger, M., Kubinger, W., and Steininger, A. (2009). SAD-based stereo matching using FPGAs. *Embedded Computer Vision*, pages 121–138.

[Ambrus et al., 2017] Ambrus, R., Claici, S., and Wendt, A. (2017). Automatic room segmentation from unstructured 3D data of indoor environments. *IEEE Robotics and Automation Letters*, 2(2):749 – 756.

[An and Ahmed, 2008] An, L. and Ahmed, S. E. (2008). Improving the performance of kurtosis estimator. *Computational Statistics & Data Analysis*, 52(5):2669 – 2681.

[ArchiCAD, 2014] ArchiCAD (2014). Graphisoft archicad bimx.

[Arias-Estrada and Xicotencatl, 2001] Arias-Estrada, M. and Xicotencatl, J. (2001). Multiple stereo matching using an extended architecture. In *Field-Programmable Logic and Applications*, pages 203–212.

[Attene et al., 2009] Attene, M., Robbiano, F., Spagnuolo, M., and Falcidieno, B. (2009). Characterization of 3D shape parts for semantic annotation. *Computer Aided Design*, 41(10):756–763.

[Baler and Allen, 2006] Baler, P. S. and Allen, P. K. (2006). Two stage view planning for large-scale site modeling. In *In Proceeding International Symposium on 3D Data Processing, Visualization and Transmission*, pages 814–821.

[Banz et al., 2010] Banz, C., Hesselbarth, S., Flatt, H., Blume, H., and Pirsch, P. (2010). Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. In *International Conference on Embedded Computer Systems (SAMOS '10)*, pages 93–101.

[Besl, 1988] Besl, P. J. (1988). Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152.

[bimen, 2015] bimen (2015). Bim energy modeling.

[Blaer and Allen, 2007] Blaer, P. S. and Allen, P. K. (2007). Data acquisition and view planning for 3D modeling tasks. *IEEE/RSJ International Conference on Intelligence Robots and Systems*, pages 417–422.

[Boehm and Becker, 2007] Boehm, J. and Becker, S. (2007). Automatic marker-free registration of terrestrial laser scans using reflectance features. *Proceedings Optical 3D Measurement Techniques*, pages 338–344.

[Boulch et al., 2013] Boulch, A., Houllier, S., Marlet, R., and Tournaire, O. (2013). Semantizing complex 3D scenes using constrained attribute grammars. *Computer Graphics Forum*, 32(5):33–42.

[Boykov and Funka-Lea, 2006] Boykov, Y. and Funka-Lea, G. (2006). Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131.

[Boykov et al., 1998] Boykov, Y., Veksler, O., and Zabih, R. (1998). Markov Random Fields with efficient approximations. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–655.

[Brown et al., 2003] Brown, M., Burschka, D., and Hager, G. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008.

[Budroni and Böhm, 2010a] Budroni, A. and Böhm, J. (2010a). Automatic 3D modelling of indoor Manhattan-world scenes from laser data. *International Journal of Architectural Computing*, 8(1):55–73.

[Budroni and Böhm, 2010b] Budroni, A. and Böhm, J. (2010b). Automatic 3D modelling of indoor Manhattan-world scenes from laser data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII:115–120.

[Calafiore, 2002] Calafiore, G. (2002). Approximation of n-dimensional data using spherical and ellipsoidal primitives. *IEEE Transactions on Systems, Man, and Cybernetics*, 32(2):269 – 278.

[Chauve et al., 2010] Chauve, A.-L., Labatut, P., and Pons, J.-P. (2010). Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 1261–1268.

[Chehata et al., 2008] Chehata, N., David, N., and Bretar, F. (2008). LIDAR data classification using hierarchical k-Means clustering. *International*

Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, XXXVII(B3b):325–330.

[Chen and Chen, 2008] Chen, J. and Chen, B. (2008). Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2-3):223–236.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean Shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

[Delaunay, 1934] Delaunay, B. (1934). Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles*, 6:793–800.

[Díaz et al., 2007] Díaz, J., Ros, E., Carrillo, R., and Prieto, A. (2007). Real-time system for high-image resolution disparity estimation. *IEEE Transactions on Image Processing*, 16(1):280–285.

[Díaz-Vilariño et al., 2014] Díaz-Vilariño, L., Martínez-Sánchez, J., Lagüela, S., Armesto, J., and Khoshelham, K. (2014). Door recognition in cluttered building interiors using imagery and lidar data. In *In Proceedings International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-5, pages 203–209.

[Diggle, 2003] Diggle, P. J. (2003). *Statistical Analysis of Spatial Point Patterns*. Edward Arnold, 2nd edition.

[Dore and Murphy, 2014] Dore, C. and Murphy, M. (2014). Semi-automatic generation of as-built BIM façade geometry from laser and image data. *Journal of Information Technology in Construction*, 19:20–46.

[Dumitru et al., 2013] Dumitru, R.-C., Borrmann, D., and Nüchter, A. (2013). Interior reconstruction using the 3D Hough transform. In *In Proceedings International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-5/W1, pages 65–72.

[Edelsbrunner and Mucke, 1994] Edelsbrunner, H. and Mucke, E. P. (1994). Three-dimensional Alpha Shapes. *ACM Transactions on Graphics*, 13(1):43–72.

[Edelsbrunner et al., 1986] Edelsbrunner, H., O'Rourke, J., and Seidel, R. (1986). Constructing arrangements of lines and hyperplanes with application. *SIAM Journal on Computing*, 15(2):341–363.

[El-Hakim et al., 1995] El-Hakim, S. F., Beraldin, J. A., and Blais, F. (1995). Comparative evaluation of the performance of passive and active 3D vision systems. In *Digital Photogrammetry and Remote Sensing*, volume 2646.

[Elseberg et al., 2012] Elseberg, J., Magnenat, S., Siegwart, R., and Nüchter, A. (2012). Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics*, 3:2–12.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proceedings International Conference on Knowledge Discovery and Data Mining*, pages 226–231.

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

[Feng et al., 2014] Feng, C., Taguchi, Y., and Kamat, V. R. (2014). Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *IEEE International Conference on Robotics and Automation*, pages 6218–6225.

[Frome et al., 2006] Frome, A., Mali, Y., and Malk, J. (2006). Image retrieval and recognition using local distance functions. In *In Proceedings Advances in Neural Information Processing Systems 19*, volume 19, pages 417–424.

[Funkhouser et al., 2003] Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., and Jacobs, D. (2003). A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105.

[Furukawa et al., 2009] Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-World stereo. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429.

[Gall et al., 2011] Gall, J., Yao, A., Razavi, N., Van Gool, L., and Lempitsky, V. (2011). Hough Forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188 – 2202.

[Gander et al., 1994] Gander, W., Golub, G. H., and Strebel, R. (1994). Least-Squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578.

[Gielsdorf et al., 2004] Gielsdorf, F., Rietdorf, A., and Gruendig, L. (2004). A concept for the calibration of terrestrial laser scanners. *TS26 Positioning and Measurement Technologies and Practices II-Laser Scanning and Photogrammetry*, pages 1–10.

[Giryes et al., 2008] Giryes, R., Bronstein, E. M., Moshe, Y., and Bronstein, M. M. (2008). Embedded system for 3D shape reconstruction. In *European DSP Education and Research Symposium*, pages 265–272.

[Gong and Yang, 2005] Gong, M. and Yang, Y.-H. (2005). Fast unambiguous stereo matching using reliability-based dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):998–1003.

[Gruen and Akca, 2005] Gruen, A. and Akca, D. (2005). Least Squares 3D surface and curve matching. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(3):151–174.

[Gudis et al., 2012] Gudis, E., van der Wal, G., Kuthirummal, S., and Chai, S. (2012). Multi-resolution real-time dense stereo vision processing in FPGA. In *International Symposium on Field-Programmable Custom Computing Machines*, pages 29–32.

[Güngör and Özmen, 2017] Güngör, E. and Özmen, A. (2017). Distance and density based clustering algorithm using Gaussian kernel. *Expert Systems with Applications*, 69:10–20.

[Hadjitheophanous et al., 2010] Hadjitheophanous, S., Ttofis, C., Georghiades, A., and Theocharides, T. (2010). Towards hardware stereoscopic 3D reconstruction a real-time FPGA computation of the disparity map. In *In Proceedings Design, Automation & Test in Europe Conference & Exhibition*, pages 1743–1748.

[Hariyama et al., 2005] Hariyama, M., Kobayashi, Y., Sasaki, H., and Kameyama, M. (2005). FPGA implementation of a stereo matching processor based on window-parallel-and-pixel-parallel architecture. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(12):3516–3522.

[Herbort and Wöhler, 2011] Herbort, S. and Wöhler, C. (2011). An introduction to image-based 3D surface reconstruction and a survey of photometric stereo methods. *3D Research*, 2(3):1–17.

[Heritage and Large, 2009] Heritage, G. L. and Large, A. R. G., editors (2009). *Laser Scanning for the Environmental Sciences*. Blackwell Publishing Ltd.

[Hile and Zheng, 2004] Hile, H. and Zheng, C. (2004). Stereo video processing for depth map. Technical report, Technical Report, University of Washington.

[Hinneburg and Keim, 1998] Hinneburg, A. and Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. In *In Proceedings International Conference on Knowledge Discovery and Data Mining*, pages 58–65.

[Hirschmuller, 2001] Hirschmuller, H. (2001). Improvements in real-time correlation-based stereo vision. In *In Proceedings Stereo and Multi-Baseline Vision*, pages 141–148.

[Hirschmuller, 2008] Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341.

[Huber and Ronchetti, 2009] Huber, P. J. and Ronchetti, E. M. (2009). *Robust Statistics*. Wiley, 2nd edition.

[Ikehata et al., 2015] Ikehata, S., Yan, H., and Furukawa, Y. (2015). Structured indoor modeling. In *In Proceedings International Conference on Computer Vision*.

[Jia et al., 2004] Jia, Y., Zhang, X., Li, M., and An, L. (2004). A miniature stereo vision machine (MSVM-III) for dense disparity mapping. In *In Proceedings International Conference on Pattern Recognition*, volume 1, pages 728–731.

[Jin et al., 2010] Jin, S., Cho, J., Dai Pham, X., Lee, K., Park, S., Kim, M., and Jeon, J. (2010). FPGA design and implementation of a real-time stereo vision system. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(1):15–26.

[Kaller et al., 2016] Kaller, O., Bolecek, L., Polak, L., and Kratochvil, T. (2016). Depth map improvement by combining passive and active scanning methods. *Radioengineering*, 25:536–547.

[Kotthauser et al., 2013] Kotthauser, T., Divband Soorati, M., and Mertsching, B. (2013). Automatic reconstruction of polygonal room models from 3D point clouds. In *IEEE International Conference on Robotics and Biomimetics*, pages 661–667.

[Kuhn et al., 2003] Kuhn, M., Moser, S., Isler, O., Gurkaynak, F., Burg, A., Felber, N., Kaeslin, H., and Fichtner, W. (2003). Efficient ASIC implementation of a real-time depth mapping stereo vision system. In *IEEE Midwest Symposium on Circuits and Systems*, volume 3, pages 1478–1481.

[Lafarge and Alliez, 2013] Lafarge, F. and Alliez, P. (2013). Surface reconstruction through point set structuring. In *In Proceedings Eurographics*, pages 5–17.

[Lalonde et al., 2006] Lalonde, J.-F., Vandapel, N., Huber, D., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861.

[Leberl et al., 2010] Leberl, F., Irschara, A., Pock, T., Meixner, P., Gruber, M., Scholz, S., and Wiechert, A. (2010). Point clouds: Lidar versus 3D vision. *Photogrammetric Engineering and Remote Sensing*, 76:1123–1134.

[Lee et al., 2005] Lee, S., Yi, J., and Kim, J. (2005). Real-time stereo vision on a reconfigurable system. *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 225–236.

[Leibe et al., 2004] Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *Conference on Computer Vision Workshop on statistical learning in computer vision*, pages 17–32.

[Lim et al., 2004] Lim, S.-N., Mittal, A., Davis, L., and Paragios, N. (2004). Uncalibrated stereo rectification for automatic 3d surveillance. In *IEEE International Conference on Image Processing*, volume 2, pages 1357–1360.

[Lu et al., 2003] Lu, C.-T., Chen, D., and Kou, Y. (2003). Algorithms for spatial outlier detection. In *In Proceedings IEEE International Conference on Data Mining*, pages 597–601.

[Lu et al., 2007] Lu, J., Rogmans, S., Lafruit, G., and Catthoor, F. (2007). High-speed dense stereo via directional center-biased windows on graphics hardware. In *3DTV Conference*, pages 1–4.

[Macher et al., 2017] Macher, H., Landes, T., and Grussenmeyer, P. (2017). From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings. *Applied Sciences*, 7(10):1030.

[Masrani and MacLean, 2006] Masrani, D. and MacLean, W. (2006). A real-time large disparity range stereo-system using FPGAs. In *In Proceedings International Conference on Computer Vision Systems*, pages 13–13.

[Meiss, 2007] Meiss, J. (2007). *Differential Dynamical Systems (Monographs on Mathematical Modeling and Computation)*. Society for Industrial and Applied Mathematics.

[Michailidis and Pajarola, 2015] Michailidis, G.-T. and Pajarola, R. (2015). Automatic reconstruction of wall features under clutter and occlusion. In *In Proceedings Computer Graphics International*.

[Michailidis and Pajarola, 2017] Michailidis, G.-T. and Pajarola, R. (2017). Bayesian graph-cut optimization for wall surfaces reconstruction in indoor environments. *The Visual Computer*, 33(10):1347–1355.

[Michailidis et al., 2014] Michailidis, G.-T., Pajarola, R., and Andreadis, I. (2014). High performance stereo system for dense 3-D reconstruction. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(6):929–941.

[Miyajima and Maruyama, 2003] Miyajima, Y. and Maruyama, T. (2003). A real-time stereo vision system with FPGA. *Field Programmable Logic And Application*, pages 448–457.

[Mozos et al., 2005] Mozos, O., Stachniss, C., and Burgard, W. (2005). Supervised learning of places from range data using adaboost. In *International Conference on Robotics and Automation*, pages 1730–1735.

[Mueller et al., 2012] Mueller, R., Teubner, J., and Alonso, G. (2012). Sorting networks on FPGAs. *The VLDB Journal*, 21(1):1–23.

[Mura et al., 2014] Mura, C., Mattausch, O., Jaspe Villanueva, A., Gobbetti, E., and Pajarola, R. (2014). Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, to appear.

[Mura et al., 2016] Mura, C., Mattausch, O., and Pajarola, R. (2016). Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. *Computer Graphics Forum*, 35(7):179–188.

[Nalpantidis et al., 2008] Nalpantidis, L., Sirakoulis, G. C., and Gasteratos, A. (2008). Review of stereo vision algorithms: From software to hardware. *International Journal of Optomechatronics*, 2(4):435–462.

[Nan et al., 2010] Nan, L., Sharf, A., Zhang, H., Cohen-Or, D., and Chen, B. (2010). SmartBoxes for interactive urban reconstruction. *ACM Transactions on Graphics*, 29(4):93:1–10.

[Nurunnabi et al., 2017] Nurunnabi, A., Sadahiro, Y., and Lindenbergh, R. (2017). Robust cylinder fitting in three-dimensional point cloud data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1/W1:63–70.

[Ochmann et al., 2016] Ochmann, S., Vock, R., Wessel, R., and Klein, R. (2016). Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, 54:94–103.

[Ochmann et al., 2014] Ochmann, S., Vock, R., Wessel, R., Tamke, M., and Klein, R. (2014). Automatic generation of structural building descriptions from 3D point cloud scans. In *In Proceedings International Conference on Computer Graphics Theory and Applications*.

[Oesau et al., 2013] Oesau, S., Lafarge, F., and Alliez, P. (2013). Indoor scene reconstruction using primitive-driven space partitioning and graph-cut. In *In Proceedings Eurographics Workshop on Urban Data Modeling and Visualisation*, pages 9–12.

[Oesau et al., 2014] Oesau, S., Lafarge, F., and Alliez, P. (2014). Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90:68–82.

[Ok et al., 2012] Ok, A. O., Wegner, J. D., Heipke, C., Rottensteiner, F., Soergel, U., and Toprak, V. (2012). Matching of straight line segments from aerial stereo images of urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 74:133–152.

[Okorn et al., 2010] Okorn, B., Xiong, X., Akinci, B., and Huber, D. (2010). Toward automated modeling of floor plans. e-Proceedings Symposium on 3D Data Processing, Visualization, and Transmission.

[Pearson, 2001] Pearson, R. K. (2001). Exploring process data. *Journal of Process Control*, 11(2):179–194.

[Pearson, 2002] Pearson, R. K. (2002). Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, 10(1):55–63.

[Pérez et al., 2009] Pérez, J., Sanchez, P., and Martinez, M. (2009). High memory throughput FPGA architecture for high-definition Belief-Propagation stereo matching. In *International Conference on Signals, Circuits and Systems*, pages 1–6.

[Pham and Jeon, 2013] Pham, C. C. and Jeon, J. W. (2013). Domain transformation-based efficient cost aggregation for local stereo matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(7):1119–1130.

[Poppinga et al., 2008] Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. (2008). Fast plane detection and polygonalization in noisy 3D range images.

In *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3378–3383.

[Pratt, 1987] Pratt, V. (1987). Direct Least-Squares fitting of algebraic surfaces. *SIGGRAPH Computer Graphics*, 21(4):145–152.

[Previtali et al., 2014] Previtali, M., Scaioni, M., Barazzetti, L., and Brumana, R. (2014). A flexible methodology for outdoor/indoor building reconstruction from occluded point clouds. *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):119–126.

[Psarakis and Evangelidis, 2005] Psarakis, E. and Evangelidis, G. (2005). An enhanced correlation-based method for stereo correspondence with subpixel accuracy. *In Proceedings IEEE International Conference on Computer Vision*, 1:907–912.

[Pu and Vosselman, 2009] Pu, S. and Vosselman, G. (2009). Building facade reconstruction by fusing terrestrial laser points and images. *Sensors*, 9:4525–4542.

[Rabbani et al., 2007] Rabbani, T., Dijkman, S., van den Heuvel, F., and Vosselman, G. (2007). An integrated approach for modelling and global registration of point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(6):355–370.

[Rousseeuw and Croux, 1993] Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283.

[Sabihuddin et al., 2008] Sabihuddin, S., Islam, J., and MacLean, W. (2008). Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array. In *Canadian Conference on Electrical and Computer Engineering (CCECE '08)*, pages 1461–1466.

[Sanchez and Zakhor, 2012] Sanchez, V. and Zakhor, A. (2012). Planar 3D modeling of building interiors from point cloud data. In *In Proceedings Internation Conference on Image Processing*, pages 1777–1780.

[Sansoni et al., 2009] Sansoni, G., Trebeschi, M., and Docchio, F. (2009). State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, pages 568–601.

[Šára, 2002] Šára, R. (2002). Finding the largest unambiguous component of stereo matching. In *In Proceedings European Conference on Computer Vision*, pages 900–914.

[Scharstein and Szeliski, 2002] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42.

[Schenk and Hanke, 2012] Schenk, S. and Hanke, K. (2012). Automatic registration of laser scanner point clouds with genetic algorithms. *Österreichische Zeitschrift für Vermessung und Geoinformation*, 99:162–170.

[Schnabel et al., 2007] Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226.

[Shao et al., 2012] Shao, T., Xu, W., Zhou, K., Wang, J., Li, D., and Guo, B. (2012). An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Transactions on Graphics*, page to appear.

[Sharma et al., 2011] Sharma, K., Jeong, K.-j., and Kim, S.-G. (2011). Vision based autonomous vehicle navigation with self-organizing map feature matching technique. In *International Conference on Control, Automation and Systems*, pages 946–949.

[Shi et al., 2011] Shi, B.-Q., Liang, J., and Liu, Q. (2011). Adaptive simplification of point cloud using k-Means clustering. *Computer-Aided Design*, 43(8):910–922.

[Silberman and Fergus, 2011] Silberman, N. and Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. In *In Proceedings International Conference on Computer Vision Workshops*, pages 601–608.

[Sousa et al., 2007] Sousa, P., Araujo, R., and Nunes, U. (2007). Real-time labeling of places using Support Vector Machines. In *International Symposium on Industrial Electronics*, pages 2022–2027.

[Stambler and Huber, 2014] Stambler, A. and Huber, D. (2014). Building modeling through enclosure reasoning. In *International Conference on 3D Vision*, volume 2, pages 118–125.

[Stamos and Allen, 2000] Stamos, I. and Allen, P. K. (2000). 3-D model construction using range and image data. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 531–536.

[Steadman, 2006] Steadman, P. (2006). Why are most buildings rectangular? *Architectural Research Quarterly*, 10(2):119–130.

[Steinhage et al., 2013] Steinhage, V., Behley, J., Meisel, S., and Cremers, A. B. (2013). Reconstruction by components for automated updating of 3D city models. *Applied Geomatics*, 5(4):285–298.

[Strecha et al., 2008] Strecha, C., von Hansen, W., Van Gool, L., Fua, P., and Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*.

[Strecha et al., 2007] Strecha, C., von Hansen, W., Van Gool, L., and Thoennessen, U. (2007). Multi-view stereo and lidar for outdoor scene modeling. In *In Proceedings Photogrammetric Image Analysis - International archives of photogrammetry, remote sensing and spatial information sciences*, volume 36, pages 167–172.

[Sui et al., 2016] Sui, W., Wang, L., and Fan, B. (2016). Layer-wise floorplan extraction for automatic urban building reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1261–1277.

[Swadzba et al., 2007] Swadzba, A., Liu, B., Penne, J., Jesorsky, O., and Kompe, R. (2007). A comprehensive system for 3D modeling from range images acquired from a 3D ToF sensor. In *In Proceedings International Conference on Computer Vision Systems*.

[Tang et al., 2010] Tang, P., Huber, D., Akinci, B., Lipman, R., and Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction*, 19(7):829–843.

[Tanskanen et al., 2013] Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., and Pollefeys, M. (2013). Live metric 3D reconstruction on mobile phones. In *In Proceedings International Conference on Computer Vision*, pages 65–72.

[Tarsha-Kurdi et al., 2007] Tarsha-Kurdi, F., Landes, T., and Grussenmeyer, P. (2007). Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from lidar data. In *In Proceedings ISPRS Workshop on Laser Scanning and SilviLaser*, pages 407–412.

[Turner et al., 2015] Turner, E., Cheng, P., and Zakhor, A. (2015). Fast, automated, scalable generation of textured 3D models of indoor environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(3):409–421.

[Turner and Zakhor, 2012] Turner, E. and Zakhor, A. (2012). Watertight as-built architectural floor plans generated from laser range data. In *In Proceedings Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 316–323.

[Turner and Zakhor, 2014] Turner, E. and Zakhor, A. (2014). Floor plan generation and room labeling of indoor environments from laser range data. In *In Proceedings International Conference on Computer Graphics Theory and Applications*.

[Valero et al., 2012] Valero, E., Adán, A., and Cerrada, C. (2012). Automatic method for building indoor boundary models from dense point clouds collected by laser scanners. *Sensors*, 12:16099–16115.

[Vanegas et al., 2012] Vanegas, C. A., Aliaga, D. G., and Benes, B. (2012). Automatic extraction of Manhattan-world building masses from 3D laser range scans. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1627–1637.

[Veksler, 2003] Veksler, O. (2003). Extracting dense features for visual correspondence with graph cuts. *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 1:689–694.

[Veksler, 2005] Veksler, O. (2005). Stereo correspondence by dynamic programming on a tree. *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2:384–390.

[Vicente et al., 2008] Vicente, S., Kolmogorov, V., and Rother, C. (2008). Graph cut based image segmentation with connectivity priors. *IEEE Computer Vision and Pattern Recognition*, pages 1–8.

[Volk et al., 2014] Volk, R., Stengel, J., and Schultmann, F. (2014). Building information modeling (BIM) for existing buildings — literature review and future needs. *Automation in Construction*, 38:109 – 127.

[Von Hansen, 2006] Von Hansen, W. (2006). Robust automatic marker-free registration of terrestrial scan data. In *Fraunhofer FOM*.

[Von Neumann, 1951] Von Neumann, J. (1951). *The General and Logical Theory of Automata, The Hixon Symposium*. John Wiley and Sons, New York.

[Vosselman et al., 2004] Vosselman, G., Gorte, B., Sithole, G., and Rabbani, T. (2004). Recognising structure in laser scanner point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 46:33–38.

[Wang et al., 2006] Wang, L., Yuan, B., and Chen, J. (2006). Robust fuzzy C-Means and bilateral point clouds denoising. In *International Conference on Signal Processing*, volume 2.

[Wei et al., 2007] Wei, S., Chen, Y., Jing, L., Atikah, S., and Ismail, J. T. (2007). Rebuilding the 3D models of buildings based on lidar data. In *In Proceedings Asian Conference on Remote Sensing*, volume I, pages 995–1003.

[Woodfill et al., 2004] Woodfill, J., Gordon, G., and Buck, R. (2004). Tyzx deepsea high speed stereo vision system. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pages 41–41.

[Woodford et al., 2014] Woodford, O. J., Pham, M.-T., Maki, A., Perbet, F., and Stenger, B. (2014). Demisting the Hough transform for 3D shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341.

[Xiong et al., 2013] Xiong, X., Adan, A., Akinci, B., and Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31:325–337.

[Xuehan et al., 2013] Xuehan, X., Antonio, A., Burcu, A., and Daniel, H. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31:325–337.

[Yang et al., 1993] Yang, Y., Yuille, A., and Lu, J. (1993). Local, global, and multilevel stereo matching. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 274–279.

[Yu et al., 2010] Yu, J., Haipeng, Z., R. Kulkarni, S., and Poor, H. V. (2010). Two-stage outlier elimination for robust curve and surface fitting. *EURASIP Journal on Advances in Signal Processing*.

[Zhan and Liang, 2011] Zhan, Q. and Liang, Y. (2011). Objects classification from laser scanning data based on multi-class Support Vector Machine. In *In Proceedings International Conference on Remote Sensing, Environment and Transportation Engineering*.

[Zhang and Zakhor, 2014] Zhang, R. and Zakhor, A. (2014). Automatic identification of window regions on indoor point clouds using LiDAR and cameras. *IEEE Applications of Computer Vision*, pages 107–114.

[Zhu and Leow, 2013] Zhu, C. and Leow, W. K. (2013). Textured mesh surface reconstruction of large buildings with multi-view stereo. *The Visual Computer*, 29(6):609–615.